Tecnológico de Estudios Superiores de Jocotitlán

Ingeniería en Sistemas Computacionales

Tema de investigación:

***Asistente basado en procesamiento del Lenguaje Natural para aprendizaje del lenguaje de programación Java***

Presentado por:

*Maldonado García Ángel*

*Ponce Camacho Julio*

Asesor Metodológico:

M. en T.C. Erika López González

Jocotitlán, México

Agradecimientos

Para nuestros familiares que nos apoyaron en el transcurso de todos nuestros estudios, impulsándonos a seguir cada día a cumplir cada uno de nuestros objetivos, brindándonos apoyo cada que estábamos abajo. Este trabajo no es solo reflejo de lo que se aprendió en la carrera, si no también es el resultado de las aptitudes y habilidades aprendidas desde el núcleo de nuestras vidas, la familia.

Resumen

El tiempo es algo fundamental en cualquier sector, y en el caso de la programación es muy importante, sin embargo, en el momento en que un estudiante empieza a aprender un lenguaje de programación, pueden surgir diversas dudas que se busquen solucionar a través de internet, pero esto puede ser complejo debido a la gran cantidad de sitios existentes en la red, confundiendo al usuario y provocando una gran inversión de tiempo que pudo haberse utilizado para la codificación del software. Por tal motivo, el objetivo de esta investigación es la creación de un asistente basado en el procesamiento del lenguaje Natural para la consulta de información, específicamente del lenguaje de programación Java, con el fin de hacer más eficiente el tiempo en la consulta de información ya que se considera que a partir de la implementación del motor de voz a texto y el posterior procesamiento del lenguaje natural, se realizara la búsqueda en la web obteniendo resultados relevantes con un 70% de precisión a la petición original recibida por el usuario.

Al ser la meta desarrollar un software, la forma en que se realizara es atreves de una metodología de software, en este caso una metodología ágil llamada SCRUM, ya que gracias a esta se puede desarrollar software rápido y determinar los requerimientos de manera constante, manteniendo un buen ritmo en el diseño, implementación y la corrección de errores.

**Índice**

[Introducción 1](#_Toc116630229)

[I. Generalidades del Proyecto 3](#_Toc116630230)

[1.1 Planteamiento del problema 3](#_Toc116630231)

[1.2 Hipótesis 5](#_Toc116630232)

[1.3 Objetivos 6](#_Toc116630233)

[1.3.1 Objetivo General 6](#_Toc116630234)

[1.3.2 Objetivos Particulares 6](#_Toc116630235)

[1.4 Justificación 7](#_Toc116630236)

[1.5 Alcances y limitaciones 9](#_Toc116630237)

[1.5.1 Alcances 9](#_Toc116630238)

[1.5.2 Limitaciones 9](#_Toc116630239)

[II. MARCO TEORICO 10](#_Toc116630240)

[2.1 Antecedentes 10](#_Toc116630241)

[2.2 Generalidades de la programación por computadora 17](#_Toc116630242)

[2.2.1 Algoritmos 17](#_Toc116630243)

[2.2.2 Lenguaje de Programación 17](#_Toc116630244)

[2.2.3 Programación Orientada a Objetos 18](#_Toc116630245)

[2.2.4 Ficheros 18](#_Toc116630246)

[2.3 Web Scraping 19](#_Toc116630247)

[2.4 Inteligencia artificial 19](#_Toc116630248)

[2.4.1 Redes neuronales artificiales. 19](#_Toc116630249)

[2.5 Procesamiento de Lenguaje Natural 21](#_Toc116630250)

[2.5.1 Reconocimiento de Voz 22](#_Toc116630251)

[2.5.2 Síntesis de Voz 23](#_Toc116630252)

[2.5.3 Softwares útiles para el procesamiento de lenguaje natural. 25](#_Toc116630253)

[III. Metodología 27](#_Toc116630254)

[3.2 Requisitos del software 32](#_Toc116630255)

[3.2.1 Requisitos funcionales 32](#_Toc116630256)

[3.2.2 Requisitos no funcionales 33](#_Toc116630257)

[3.2.3 Diagrama de casos de uso 34](#_Toc116630258)

[3.2.4 Actores en el sistema 35](#_Toc116630259)

[3.2.5 Casos de uso 36](#_Toc116630260)

[3.3 Planeación Sprint 42](#_Toc116630261)

[3.4 Sprint 44](#_Toc116630262)

[3.4.1 Diseño del software 44](#_Toc116630263)

[3.4.2 Implementación 56](#_Toc116630264)

[3.4.3 Pruebas 67](#_Toc116630265)

[IV. Resultados obtenidos y discusión 70](#_Toc116630266)

[V. Conclusiones 72](#_Toc116630267)

[Fuentes de información 73](#_Toc116630268)

[Anexos 76](#_Toc116630269)

Índice de figuras

[Figura 1 Etapas del reconocimiento de voz 22](#_Toc106568851)

[Figura 2 Diagrama metodología basada en SCRUM 27](#_Toc106568852)

[Figura 3 Fases dentro del Sprint 29](#_Toc106568853)

[Figura 4 Cronograma de Actividades 30](#_Toc106568854)

[Figura 5 Diagrama de casos de uso 35](#_Toc106568855)

[Figura 6 Diseño lógico del software 45](#_Toc106568856)

[Figura 7 Diseño ventana principal 46](#_Toc106568857)

[Figura 8 Diseño Interfaz de entrada por voz 47](#_Toc106568858)

[Figura 9 Diseño entrada por teclado 47](#_Toc106568859)

[Figura 10 Diseño de ventana para el despliegue de resultados 48](#_Toc106568860)

[Figura 11 Diseño ventana inserción de código local 48](#_Toc106568861)

[Figura 12 Diseño interfaz de configuración de software 49](#_Toc106568862)

[Figura 13 Modelo vista controlador (MVC) 50](#_Toc106568863)

[Figura 14 Ficheros del asistente 53](#_Toc106568864)

[Figura 15 Formato del fichero local 54](#_Toc106568865)

[Figura 16 Diagrama de clases 56](#_Toc106568866)

[Figura 17 Diagrama de flujo del funcionamiento General 58](#_Toc106568867)

[Figura 18 Diagrama de flujo del funcionamiento del reconocimiento de voz 60](#_Toc106568868)

[Figura 19 Flujo de instrucciones de la síntesis de voz 61](#_Toc106568869)

[Figura 20 Manejo de inserción de código local 61](#_Toc106568870)

[Figura 21 Algoritmo de Web Scraping 63](#_Toc106568871)

[Figura 22 Codificación de la interfaz Gráfica 64](#_Toc106568872)

[Figura 23 Formato de códigos en la interfaz 65](#_Toc106568873)

[Figura 24 Implementación de hilos en el programa 66](#_Toc106568874)

[Figura 25 Implementación del software 67](#_Toc106568875)

[Figura 26 Consumo en repeso del software 69](#_Toc106568876)

[Figura 27 Consumo al ejecutar petición 69](#_Toc106568877)

[Figura 28 Consumo al realizar procesamiento de voz 69](#_Toc106568878)

[Figura 29 Pruebas de la precisión de búsqueda web 70](#_Toc106568879)

[Figura 30 Pruebas de reconocimiento de voz con condiciones optimas 70](#_Toc106568880)

[Figura 31 Pruebas de reconocimiento de voz en condiciones de ruido en el ambiente 71](#_Toc106568881)

[Figura 32 Pruebas de reconocimiento de voz con petición dictada a gran velocidad 71](#_Toc106568882)

Índice de Tablas

[Tabla 1 Tipos de aprendizaje de las redes neuronales.[19] 20](#_Toc106568883)

[Tabla 2 Tipos de reconocimiento de voz. [21] 23](#_Toc106568884)

[Tabla 3 Tipos de síntesis de voz [22] 24](#_Toc106568885)

[Tabla 4 Herramientas para el procesamiento del lenguaje natural [23] 25](#_Toc106568886)

[Tabla 5 Requisitos funcionales 32](#_Toc106568887)

[Tabla 6 Requisitos no funcionales 33](#_Toc106568888)

[Tabla 7 Tabla de actores del sistema: Usuario 36](#_Toc106568889)

[Tabla 8 Caso de uso: Iniciar aplicación 36](#_Toc106568890)

[Tabla 9 Caso de uso: Registrar Código Local 36](#_Toc106568891)

[Tabla 10 Caso de uso: Solicitud a través de voz 37](#_Toc106568892)

[Tabla 11 Caso de uso: Solicitud a través del teclado 38](#_Toc106568893)

[Tabla 12 Caso de uso: Búsqueda local 38](#_Toc106568894)

[Tabla 13 Caso de uso: Búsqueda en la web 39](#_Toc106568895)

[Tabla 14 Caso de uso: Respuesta de solicitud 39](#_Toc106568896)

[Tabla 15 Caso de uso: Ir a siguiente coincidencia 40](#_Toc106568897)

[Tabla 16 Caso de uso: Cambio de configuración 40](#_Toc106568898)

[Tabla 17 Caso de uso: Cambio de color de interfaz 41](#_Toc106568899)

[Tabla 18 Caso de uso: Cambio de velocidad de síntesis de voz 41](#_Toc106568900)

[Tabla 19 Caso de uso: Cambio de modo de visualización 42](#_Toc106568901)

[Tabla 20 Planeación de Sprints para desarrollar el proyecto 43](#_Toc106568902)

[Tabla 21 Dependencias necesarias para Python 57](#_Toc106568903)

[Tabla 22 Pruebas de búsqueda web con respecto a sintaxis de funciones 67](#_Toc106568904)

[Tabla 23 Pruebas de búsqueda Web con respecto a definiciones 68](#_Toc106568905)

# Introducción

Los asistentes inteligentes son una tecnología que en la actualidad se ha vuelto muy popular, debido a las múltiples funciones con las que cuentan, y de algún modo permiten una interacción natural entre los dispositivos y los seres humanos, “Los asistentes virtuales son herramientas inteligentes que ayudan a los usuarios a buscar información en un conglomerado de recursos web” [13]. La educación es un factor muy importante en el desarrollo de una persona, por ende, la relación entre la tecnología y la educación son dos aspectos que deberían de manejarse de manera conjunta para un buen desempeño en el aprendizaje de diversos temas.

Los asistentes inteligentes en la actualidad pueden tener múltiples tipos de usos, normalmente están enfocados en el uso de propósito general, o funciones específicas, pero en cuestiones de educación no se encuentra tan implementada este tipo de tecnología. El objetivo de desarrollar este proyecto de investigación, es el implementar un asistente basado en PLN (Procesamiento de lenguaje natural) dedicado al aprendizaje del lenguaje de programación Java, contando con la característica de orientar al usuario sobre el uso de este lenguaje a partir de fuentes confiables y validadas, fomentando las bases para un buen desarrollo en el aprendizaje a través del reconocimiento de voz y síntesis de voz por parte del asistente, permitiendo la comunicación entre el software y el usuario a través de lenguaje natural. Uno de los principales usos de este asistente es la de instruir o ayudar al usuario resolviendo dudas como, por ejemplo, en la sintaxis de alguna función o determinando la estructura de diversos comandos que están documentados en sitios confiables para el estudio del lenguaje de programación.

Se considera que el tema de los asistentes inteligentes es de gran relevancia en la actualidad, y probablemente tengan un impacto aun mayor en un futuro, y el ser parte de ese desarrollo es lo que motiva la realización de esta investigación, con el fin de implementar la creación de este tipo de software.

El tema de los asistentes inteligentes ha sido un tema de mucho interés investigativo con anterioridad, por lo que muchas personas y organizaciones han realizado diversos proyectos relacionados con la creación de tecnologías de esta índole, obteniendo diversos tipos de resultados, algunos alcanzando la meta planteada y algunos otros que dejaron inconcluso el desarrollo del proyecto por problemas que pudieron surgir en su ejecución; analizar los logros y fracasos de estos proyectos predecesores pueden ser una oportunidad para implementar o mejorar diversos aspectos en el proyecto sobre un asistente inteligente que se plantea realizar.

# I. Generalidades del Proyecto

## 1.1 Planteamiento del problema

Dado el contexto de un estudiante que comienza en el estudio de un lenguaje de programación, pueden surgir diversas dificultades que dan como resultado un aprendizaje lento y una comprensión del lenguaje baja. Una de estas adversidades a las que se enfrenta un programador novato es la de desconocer la sintaxis utilizada en un lenguaje de programación de alto nivel, como lo es Java, debido a que existe gran cantidad de información que involucra la inversión de alta cantidad de tiempo para determinar cuáles son los datos que se adecúan a lo que se está buscando, contando con una baja precisión en la obtención de los datos necesitados, debido a que en la actualidad la red está repleta de información, pero esto en lugar de tomarse como una ventaja se puede convertir en una desventaja debido a la gran cantidad de la misma, llegando a causar confusión al lector de la información, debido a que entre todos los sitios existentes se puede perder fácilmente entre tantas fuentes, dando como consecuencia una inversión de tiempo alta.

El tiempo es el recurso más valioso en la vida de una persona, por esto mismo es importante aprovecharlo lo más que se pueda. Un programador debe de saber aprovechar su tiempo correctamente para enfocarlo a la resolución de problemas, y es que cuando se realiza la codificación de un software, pueden surgir diversas situaciones en las que se requiere de una consulta en la red para poder seguir avanzando, lo cual puede tomar mucho tiempo en la búsqueda de la solución, y es que entre la gran cantidad de sitios que hay es tardado analizar la información de cada uno, y en el caso de un novato es aún peor. Otro factor que aumenta el tiempo de la búsqueda de algún aspecto es la velocidad que el usuario tiene para formular y escribir la consulta, ya que escribir no es lo mismo que hablarlo directamente, causando que se consuma demasiado tiempo en algo no directamente productivo, haciendo ineficiente el desarrollo de los programas y a su vez frustrando al usuario por el hecho de sentir que no avanza.

Por tales motivos, se plantea el desarrollo de un asistente basado en PLN (Procesamiento de Lenguaje Natural) que sea capaz de filtrar la información presentada al usuario final mediante la extracción de un fragmento textual que contenga la información consultada, todo ello con base a la priorización de los sitios mejor valorados estableciéndolos como repositorios principales para ser la fuente de consulta, y a su vez, el asistente contara con la capacidad de almacenar códigos propios del usuario, con el propósito de que el estudiante asimismo pueda consultar los algoritmos que en algún momento desarrolló y puedan ser de utilidad en la creación de futuros programas.

## 1.2 Hipótesis

A partir de la implementación del motor de voz a texto y el posterior procesamiento del lenguaje natural; se realizará la búsqueda en la web obteniendo resultados relevantes con un 70% de precisión a la petición original recibida por el usuario.

## 1.3 Objetivos

### 1.3.1 Objetivo General

Desarrollar un asistente basado en el procesamiento del lenguaje Natural para la consulta de información confiable del lenguaje de programación Java presentada al usuario final mediante un fragmento de texto.

### 1.3.2 Objetivos Particulares

* Investigar los antecedentes de los asistentes basados en PLN, con el fin de entender cuáles son los avances logrados en este tipo de softwares.
* Investigar los motores de conversión de texto a voz y los requerimientos de cómputo que necesitan.
* Diseñar la estructura del sistema, dando soporte a una entrada de voz y el manejo de una interfaz minimalista.
* Definir un formato de documentación para almacenar los códigos de usuario de manera local.
* Implementar un motor de conversión de voz a texto, con el propósito de lograr que el asistente comprenda lo que el usuario exprese a través del lenguaje natural.
* Implementar un sistema de identificación de palabras clave, con la finalidad de ser utilizadas para realizar la búsqueda de información.
* Crear una modulo dedicado a la búsqueda de información a través de la web con la capacidad de filtrar fuentes oficiales o validadas.
* Evaluar la funcionabilidad del asistente, aplicando diversas pruebas para comprobar si cumple el funcionamiento esperado.
* Documentar el desarrollo y resultados del proyecto.

## 1.4 Justificación

La programación en la actualidad es un factor bastante importante, ya que prácticamente todos los dispositivos digitales llevan algún tipo de programación, sea para sistemas embebidos que realizan una única función, o bien, en la programación de aplicaciones que pueden ser útiles en sistemas operativos de propósito general. El estudiar un lenguaje de programación puede ser sencillo o bastante difícil, depende mucho de las aptitudes del estudiante, sin embargo, es muy normal que una persona que esta apenas iniciando en este estudio, se encuentre con gran cantidad de adversidades que complican el aprendizaje de este tipo de lenguajes, y más cuando no se encuentran directamente orientados por una persona experimentada, incluso aunque hoy en día se cuenta con internet, puede ser bastante tedioso para un programador novato encontrar información precisa y correcta en la gran cantidad de sitios que existen, dando como consecuencia la inversión de mucho tiempo en la búsqueda de información específica, que al final del día pudo no haber sido tan útil al quedar más confundido entre toda la información de los sitios consultados.

Por tal motivo, el asistente basado en PLN para el aprendizaje del lenguaje de programación Java permitirá a los estudiantes obtener información correcta sobre las dudas que surjan en el transcurso de su aprendizaje en el lenguaje de programación, dando prioridad a las fuentes más usadas y mejor valoradas para la búsqueda de información, contando con la característica de que el usuario pueda consultar por medio del lenguaje natural cuestiones como la sintaxis de alguna función o método, el significado de algún componente en el lenguaje, la definición teórica de diversos aspectos del paradigma de programación, algoritmos concretos o recomendaciones sobre como programar correctamente. Esto con el propósito de que el estudiante tenga un buen desarrollo en el aprendizaje del lenguaje, siendo un apoyo para que el usuario pueda aclarar sus dudas rápidamente y pueda continuar aprendiendo otros aspectos de la programación.

Un factor a mencionar es que muchos programadores tienden a basar su código en otros, ya sea de autoría propia o de otra persona, pero abordando el primer aspecto, es común que muchas veces el programador desarrolle métodos o funciones que puedan ser base para futuras implementaciones, siendo una gran ventaja para ahorrar tiempo y no iniciar de cero, es por ello que el asistente de igual forma contará con la capacidad de almacenar bloques de código que el usuario pueda reutilizar en futuros proyectos, especificando un identificador, descripción y el código, para que el usuario pueda tener acceso a ellos con tan solo pedírselo al asistente, el cual realizará la búsqueda de manera local y mostrará lo que el usuario ingresó anteriormente, ahorrando tiempo y haciendo el proceso de desarrollo más eficiente.

Los principales sujetos beneficiados de este proyecto serán los estudiantes que están empezando en el mundo de la programación o cualquier persona que quiera programar en java, facilitando la consulta de información y agilizando los tiempos de desarrollo de las aplicaciones que codifiquen.

## 1.5 Alcances y limitaciones

### 1.5.1 Alcances

* El asistente podrá funcionar en computadoras de escritorio o laptops que cuenten con el sistema operativo Windows 10/11.
* El sistema podrá emplear el uso de reconocimiento de voz y de manera textual a través de la interfaz de usuario.

### 1.5.2 Limitaciones

* El asistente requiere de una conexión internet para funcionar de manera correcta.
* Los resultados de búsqueda dependerán de un conjunto de sitios seleccionados.
* Las primeras versiones del software estarán en constante aprendizaje por lo que los resultados estarán muy cerca del límite inferior de precisión (70%).

# II. MARCO TEORICO

## 2.1 Antecedentes

Los asistentes inteligentes son una tecnología que se puede implementar en muchos ámbitos, desde el desarrollo de un asistente con un funcionamiento particular, por ejemplo, ser apoyo para el manejo de algún dispositivo; hasta asistentes con el propósito de atender necesidades generales de un usuario. Esta tecnología ha estado presente desde el año 1960, por lo cual han surgido diversos trabajos que han incrementado el desarrollo de esta.

El hablar de asistentes inteligentes, obliga a abordar el tema del procesamiento del lenguaje natural, ya que estes de gran importancia para que un asistente pueda entendernos. Esto se remonta en el año 2006 en donde se propuso un nuevo modelo con el fin de aumentar la precisión del reconocimiento de voz por el autor José Oropeza, que mediante un artículo realizó un trabajo sobre “Los algoritmos y métodos para el reconocimiento de voz en español mediante sílabas “[1] el cual tenía el objetivo de crear una alternativa al paradigma del reconocimiento de voz, enfocando el funcionamiento a la detección de las silabas que constituyen las palabras.

El reconocimiento basado en sílabas fue principalmente enfocado al idioma español y consiste en establecer un sistema que esté basado en conocimientos, en la cual se clasifican las silabas del lenguaje español, para posteriormente a base de la entrada de voz se realiza el reconocimiento, concatenando cada silaba para determinar qué es lo que se está diciendo por parte del hablante, logrando una precisión del 96% en el entendimiento del habla en el lenguaje español.

Existen diversidad de autores que han abordado la implementación y mejora del modelo del reconocimiento de voz, sin embargo, en estas fechas también empezaban las primeras implementaciones más avanzadas de esta tecnología orientada a ciertos tipos de usuarios. En el artículo con el título “Asistente Inteligente basado en Agentes para modelar Sistemas mediante el uso de UML”, el autor Mauricio Paletta [2] del año 2008, se propuso un asistente inteligente útil para modelar sistemas. Con el propósito de orientar y ayudar tanto a usuarios expertos como usuarios novatos en el tema, en el desarrollo de sistemas de basado en UML.

El diseño de este agente es realizado mediante el uso de SIFIVA, un ambiente abierto para el desarrollo de sistemas basados en agentes y en la integración de conceptos diferentes e independientes tales como SOA (Arquitectura Orientada a los Servicios), FIPA (Fundación de Agentes Físicos Inteligentes), XML (Lenguaje de marcado extendible) y BDI (Bases de datos interoperantes). Los resultados muestran que un agente implementado con SIFIVA puede ser utilizado como ayuda para el modelado de sistemas haciendo uso de UML con una precisión de hasta 78.88% según una encuesta aplicada a usuarios que usaron el sistema.

Los asistentes como el basado en UML, ayudan a los usuarios a realizar tareas concretas, y así como este existen más ejemplos como en el artículo del 2010 “*un asistente inteligente para UNIX* basado en documentación”, escrito por Manuel Buenaga & Baltasar Fernández *et al* [3], se aborda el desarrollo de un asistente inteligente en línea como propuesta de alternativa a los tutores inteligentes, este asistente inteligente tiene como nombre “*ARGOS*”, el cual fue desarrollado para ser una fuente de ayuda para los usuarios que hagan uso de los sistemas operativos basados en Unix, teniendo la funcionalidad de orientarlos en aspectos como el manejo del sistema a través de comandos o especificando las diversas funciones que se pueden realizar con el sistema, esto a través de técnicas de recuperación de información a partir de una lista de afinidades léxicas, modelado de usuario para la adaptación a usuarios concretos y técnicas de hipertexto. El resultado final de este proyecto quedó inconcluso, sin embargo, el funcionamiento base se cumplió, pero el lenguaje natural que logra entender quedo algo limitado.

A parte de asistentes orientados a un propósito especifico, de igual manera se han enfocado a la enseñanza académica, como se menciona en el documento del 2011 “*Asistentes virtuales de clase en la educación universitaria”*, redactado por los autores Mauricio Dorfman & Andrea Grondona *et al* [4], abordan diversos aspectos de los asistentes inteligentes enfocados en el área educativa, como apoyo principalmente de los estudiantes universitarios, haciendo énfasis en el asistente “*Ariel”*, con un “Cerebro artificial” capaz de comprender el lenguaje español. *Ariel* hace uso de técnicas PLN (Procesamiento de lenguaje natural) para interpretar la conversación y formulando diversas respuestas con base en el entendimiento.

Los resultados de este asistente fueron favorables, teniendo módulos para síntesis de voz y la sincronización con un avatar que lo representa, contando con el conocimiento del 90% de los tópicos asociados a la asignatura de administración de recursos informáticos, incluyendo temas como planificación, selección de aplicaciones, uso estratégico de TICs, etc. Esta investigación fue la base de próximas investigaciones, tanto de artículos como de tesis.

El desarrollo de librerías o tecnologías aplicables en diversos lenguajes para el reconocimiento de voz ha estado en desarrollo constante, y diversos autores han tratado de implementar métodos propios con el fin de hacer más precisa la detección. Un ejemplo de esto es el trabajo de Guillermo Martínez & Gualberto Aguilar, quienes propusieron en un artículo en la revista INGENIUS del 2013, un sistema de reconocimiento de voz basado en MFCC (Coeﬁcientes Cepstrales en las Frecuencias de Mel), SBC (parámetros Cepstrales Basados en Sub-banda) y espectrogramas [14] con el fin de tomar ventaja de los 3 algoritmos para aumentar la precisión del reconocimiento de voz. Lo que se planteó en ese artículo, fue un sistema que aprovechara las características de los 3 algoritmos mencionados anteriormente, empleando una máquina de soporte vectorial con el propósito de clasificar, para que después cada algoritmo arroje un grupo con las probabilidades más altas de ser correspondidas para hacer una evaluación, tomando una decisión con la persona con más probabilidades en los 3 algoritmos. Los resultados del sistema no fueron los esperados, ya que su precisión era del 93.33% y se esperaba que esta cifra fuera aun mayor ya que por ejemplo el algoritmo MFCC con Gabor arroja hasta un 97.77%.

Un proyecto similar a ARGOS (proyecto mencionado anteriormente) con la diferencia de tener un listado de respuestas predefinidas fue el del autor Cristian Fernández Pardo, en el cual se propone “El desarrollo e implementación de un asistente virtual para Linux” [6] en el año 2014. El objetivo de la realización de este asistente virtual fue la de poder gestionar y manipular un sistema operativo Linux mediante el uso de diversas ordenes sencillas de voz, esto realizándose a partir de tecnologías como ¡ATROS, que tiene la función de reconocer el habla, en conjunto con Eutranscribe, el cual es un script escrito en Perl para la transcripción fonética en español. Los resultados de este proyecto fueron regulares debido a que ¡ATROS estaba algo limitado, ya que no siempre reconocía las palabras que el usuario decía, dando una tasa de error de hasta el 30%, y a su vez existía deficiencia en la cantidad de órdenes, siendo poco práctico al usarlo día a día. Estas tecnologías son de gran utilidad en cada uno de los sectores al cual está orientado, sin embargo, no únicamente se han desarrollado proyectos orientados al software, sino que igualmente han existido propuestas para revolucionar como se implementan este tipo de asistentes a manera teórica.

El autorAlíen Martin González de la Universidad de Matanzas de Cuba [7], propuso en 2016 mediante un artículo, un nuevo y revolucionario concepto en el mundo de la informática y la educación, una mezcla entre inteligencia artificial y conocimiento que busca perfeccionar el proceso de enseñanza. Lo que hizo Alíen Martín, fue abordar el tema de agentes inteligentes en objetos o entornos de aprendizaje, el artículo se basa en la recopilación de información del aprendizaje y niveles de conocimiento que debe tener el estudiante, su funcionamiento es que el agente mantiene al tanto al profesor de los avances y no es necesario que exista interacción entre el profesor y el estudiante.

Lo que logro con este artículo fue sentar la base para entender que se necesita para desarrollar un asistente de conocimiento personal en la educación de una forma sencilla de entender y lo más completa posible. Los asistentes inteligentes de igual forma se han implementado mediante chatbots, contando con el reconocimiento a partir de una entrada del teclado, siendo este también un tema de interés para muchos investigadores.

El autor Rafael Aguilar Mejía, en 2016, propuso una implementación sobre la tecnología de la inteligencia artificial, el cual mediante una Tesis aborda el desarrollo de una secuencia didáctica que integra el uso de chatbot con una actividad de aprendizaje activa enfocada solamente en el primer año de Física [8]. El resultado fue positivo, se logró integrar un chatbot con una actividad de aprendizaje activo como los tutoriales, con el propósito de analizar su impacto en el entendimiento conceptual de un tema específico y esencial de física.

Algunas otras implementaciones dentro del campo de la inteligencia artificial impulsaron el desarrollo de nuevos proyectos. El articulo realizado por Arteaga Maza *et al* [9] en el año 2017, presenta la Implementación de un asistente tutor basado en computación cognitiva para su uso en entornos visuales de aprendizaje, con el fin de mejorar los conocimientos de los estudiantes, resolviendo dudas.

El proyecto contó con solo una interfaz desarrollada a través del framework Node.js, con el servicio de conversación “IMB Watson”, donde el usuario puede ingresar los cuestionamientos que existan de manera escrita. Los resultados obtenidos por los testers fue que el asistente cumple correctamente con su función, sin embargo, el hecho de escribir cada solicitud y responder con más texto no fue muy eficiente para algunas personas.

El campo de los asistentes no solo está en lo académico o la ayuda personal, también abarca más áreas, un ejemplo es el proyecto de asistente para el hogar en español creado por Carlos Díaz Fernández [10] en el 2017. Este asistente está compuesto por un sistema que trabaja sobre una Raspberry Pi 2, mediante el cual, se pueden realizar tareas que un usuario le exija simplemente con el uso de su propia voz con el fin de que el dispositivo responda a esta petición y tome las decisiones pertinentes. El diseño del prototipo y la codificación se implementó a partir de una biblioteca de Python Speech Recognition, con el cual se logró hacer el reconocimiento de voz. La precisión de este asistente era del 88.66% según las pruebas de precisión al reconocer las órdenes, sin embargo, las peticiones que ejecutaba eran muy limitadas.

En prácticamente todos los proyectos mencionados, se desarrollan estos asistentes virtuales desde la programación con lenguajes de alto nivel, o haciendo uso de herramientas específicamente destinadas al desarrollo, sin embargo, también se pueden implementar a partir de herramientas ya existentes que ni si quiera tenían ese fin, aprovechando las diversas funcionalidades para crear un asistente, o eso es lo que se propone con el uso de Facebook como herramienta de aprendizaje colaborativo de la inteligencia artificial, por Maikel Leyva, Rebeca Escoba *et al* [11] en el 2018.

En el trabajo se analiza el impacto de Facebook, como herramienta para el aprendizaje colaborativo de la Inteligencia Artificial. Se muestran los resultados de una encuesta aplicada a los estudiantes y el engagement generados por los distintos tipos de publicaciones. Esta información coincide con lo reportado en la literatura de ser los videos lo que genera mayor engagement en los usuarios. Adicionalmente se refuerza la percepción de que Facebook genera percepciones mayormente positivas, así como un sentido de comunidad reforzando esta red social como una alternativa a los sistemas de gestión de aprendizaje online.

Han existido proyectos que tratan temas a gran escala, pero de igual manera se han implementado proyectos a un sector especifico, donde los beneficiados son una pequeña región, como ejemplo es el asistente virtual tipo chatbot, desarrollado por el autor Luis Felipe García [12] desarrollado en 2018. En este trabajo de investigación para titulación, se desarrolló un asistente virtual tipo chatbot, teniendo la característica de recibir peticiones únicamente por el teclado y dar respuestas a través de la pantalla, teniendo el propósito de hacer más rápida la recepción de una solicitud de PQR’s (El Sistema de Peticiones, Quejas, Reclamos y Sugerencias) en la corporación San Isidro “Colegio Anglo Americano”.

Para desarrollar este proyecto, se determinaron las técnicas más comunes para dar soluciones a PQR’s de primer nivel, y con base a ello se modeló la herramienta inteligente con un módulo de administración web donde se realizará la interacción entre el usuario el chatbot. Los resultados no fueron los esperados, ya que el uso de este asistente fue bueno solo en ciertas situaciones básicas, ya que existieron problemas como solicitudes que el chatbot no era capaz de procesar, terminando en un software incompleto.

El reconocimiento de voz es una tecnología muy útil para resolver dudas mediante el lenguaje natural, pero este de igual forma es útil para agilizar la ejecución de cierta acción, un ejemplo de esto es el trabajo de tesis “Sistema de Seguridad Por Reconocimiento de Voz” escrito por los autores Eyra Pérez & Fernando Martínez et al [13] en el año 2018. En este trabajo de titulación se planteó la implementación de un sistema de seguridad con la característica principal del uso del reconocimiento de voz con el fin de que el sistema pueda interactuar de una manera más fluida y precisa, para poder tomar decisiones rápidas y confiables.

El diseño e implementación de este sistema, fue realizado mediante la utilización de algoritmos integrados en un software llamado MATLAB, el cual cuenta con funciones que permiten analizar la señal capturada por un micrófono y reconocer al sujeto que está hablando, teniendo el registro previo de la voz con diversos archivos de audio. Se realizaron pruebas que en su mayoría fueron precisas, sin embargo, un factor importante para evitar errores fue estar en un ambiente de 70 dB (Decibeles), existiendo el problema de que el sistema no lograba reconocer a las personas cuando estos hablaban en un tono de alegría o de tristeza, teniendo una tasa de error del 40%, por lo que este sistema no fue óptimo.

Otra utilidad que los asistentes virtuales pueden tener se ejemplifica en el trabajo de Erick Ibarra Cruz, de la facultad de filosofía y letras [14], implementó un asistente basado en esta tecnología para ambientes de aprendizaje de niños con discapacidad visual en el 2020, tomando en cuenta las características de la voz como lo es su tono, la reacción con la que se expresa el usuario, etc. Siendo factores importantes y necesarios para el correcto funcionamiento del asistente. El fin que tiene este asistente es mejorar el aprendizaje de los niños con algún tipo de discapacidad visual. Los resultados que se obtuvieron fueron, que una tercera parte de los estudiantes tuvieron un aprendizaje mayor. Por lo que la conclusión de este proyecto no cumplió los objetivos planteados, al no manejar correctamente los diversos modos de enseñanza.

Diversos autores se han aventurado a desarrollar asistentes de una manera diferente con el fin de mejorar algún aspecto de la relación humano-máquina, como es el caso de Mónica Echeverrin & Roberto Manjarres, los cuales mediante un artículo llamado “Asistente virtual académico utilizando tecnologías cognitivas de procesamiento de lenguaje natural” [15] proponen el desarrollo de un prototipo de asistente virtual académico tipo chatbot. El prototipo se desarrolló con el fin de mejorar la experiencia y tiempos de atención en el área de procesos académicos, específicamente en el Politécnico Colombiano Jaime Isaza Cadavid. Se desarrolló con base a tecnologías como Dialogflow, para clasificar las intenciones de los usuarios y así poder contestar una respuesta, esto se implementó en la plataforma de WhatsApp a partir del framework Node.js y PostgreSQL para almacenar la información. Una ventaja de este prototipo es que con Dialogflow da la posibilidad de migrar a diversos sitios donde los estudiantes pueden acceder. Los resultados del asistente fueron satisfactorios por parte de los autores, ya que con base a encuestas se pudo determinar la satisfacción de los usuarios la cual fue del 80%.

Todos estos proyectos realizados han tenido diversos tipos de enfoques, desde un asistente para materias específicas en el estudio de algún sector, hasta asistentes para facilitar el proceso de la realización de alguna actividad, como lo es el uso de un sistema operativo o automatizar diversas tareas en casa. En base al análisis de estos proyectos, el panorama abarcado del tema de un asistente inteligente aumenta, por lo cual el definir que se pretende resolver en la realización de este tipo de proyectos queda aún más claro, pudiendo definir de una manera más concisa el problema y la solución que se pretenden implementar para que el proyecto a realizar tenga un sustento más completo.

## 2.2 Generalidades de la programación por computadora

### 2.2.1 Algoritmos

Se define como una secuencia de instrucciones que representan un modelo de solución para determinado tipo de problemas. O bien como un conjunto de instrucciones que realizadas en orden conducen a obtener la solución de un problema. Consiste en aplicar adecuadamente una serie de pasos detallados que aseguran una solución correcta. Por lo general, cada algoritmo es específico de un dominio del conocimiento. [16]

Los algoritmos representan la solución de un problema a través de ciertos pasos finitos, los cuales se pueden representar a través de un diagrama de flujo para una mayor comprensión gráfica, para después realizarse en un pseudocódigo, estableciendo de esta manera el mejor camino para realizar la codificación del algoritmo a través de un lenguaje de programación para su ejecución a través de una computadora.

### 2.2.2 Lenguaje de Programación

Un lenguaje de programación es un conjunto de reglas o normas que permiten asociar a cada programa correcto un cálculo que será llevado a cabo por un ordenador (sin ambigüedades). Por tanto, un lenguaje de programación es un convenio o acuerdo acerca de cómo se debe de interpretar el significado de los programas de dicho lenguaje. [16]

Un lenguaje de programación está constituido a partir de diversas reglas, estos lenguajes pueden tener un enfoco diferente en cuanto al proceso de programación para resolver un problema, a lo cual se le denomina paradigma de programación, existiendo paradigmas como el declarativo, funcional, imperativo y el paradigma de programación orientada a objetos, siendo este último uno de los más utilizados, siendo el que rige al lenguaje de programación Java.

### 2.2.3 Programación Orientada a Objetos

La Programación Orientada a Objetos (POO) es un [paradigma de programación](https://profile.es/blog/que-son-los-paradigmas-de-programacion/), es decir, un modelo o un estilo de programación que brinda unas guías sobre cómo trabajar con él. Se basa en el concepto de clases y objetos, Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos.

Lenguajes secuenciales como COBOL o procedimentales como Basic o C, se centraban más en la lógica que en los datos. Otros más modernos como [Java](https://profile.es/blog/variables-tipos-datos-java/), C# y [Python](https://profile.es/blog/python/), utilizan paradigmas para definir los programas, siendo la Programación Orientada a Objetos la más popular la Programación Orientada a objetos permite que el código sea reutilizable, organizado y fácil de mantener, la programación orientaba objetos consta de clases, objetos e instancias. [17].

Algunos de los lenguajes anteriormente mencionados como lo son Java o Python, trabajan en diversos sectores de la industria, desde el desarrollo de aplicaciones móviles, software de escritorio, inteligencia artificial entre otros; la inteligencia artificial es uno de los temas más populares y que actualmente sigue desarrollándose, teniendo mejoras avances sobre este ámbito, siendo que aplicarla en diferentes entornos puede ser una gran ventaja.

### 2.2.4 Ficheros

Un fichero es un conjunto de información relacionada, grabada en el sistema de almacenamiento secundario y a la que se hace referencia mediante un nombre. [17]

## 2.3 Web Scraping

Web Scraping hace referencia a las técnicas que permiten recolectar u obtener datos desde la web a través de métodos automáticos o manuales. Con web Scraping se puede extraer el contenido de un HTML de sitios web para filtrar la información y almacenarla para después hacer uso de los datos de interés. [18]

## 2.4 Inteligencia artificial

Fleifel define la Inteligencia Artificial como la rama de la ciencia de la computación que estudia la resolución de problemas no algorítmicos mediante el uso de cualquier técnica de computación disponible, sin tener en cuenta la forma de razonamiento subyacente a los métodos que se apliquen para lograr esa resolución. [19]

La manera en que un sistema se pueda hacer inteligente es a través de las redes neuronales artificiales, las cuales permiten que tenga un funcionamiento autónomo para la funcionalidad que fue creado el sistema, siendo las redes neuronales un pilar en el campo de la inteligencia artificial.

### 2.4.1 Redes neuronales artificiales.

Las redes neuronales son un pilar en la inteligencia artificial. Son una forma de emular ciertas características propias de los humanos, como la capacidad de memorizar y de asociar hechos. Si se examinan con atención aquellos problemas que no pueden expresarse a través de un algoritmo, se observará que todos ellos tienen una característica en común: la experiencia. [20]

Existen diversas formas en que la red neuronal puede aprender para lograr el funcionamiento para la que fue programada, esto se puede entender a partir de los tipos de aprendizaje, observados en la tabla 1.

El determinar el tipo de aprendizaje con el que contara una red neuronal es de gran importancia ya que, gracias a esto, es que la red neuronal va a desarrollarse y de esta manera lograr aprender y dar la función para la que fue realidad. En esta investigación el tipo de red neuronal a utilizar es de tipo aprendizaje supervisado.

Tabla 1 Tipos de aprendizaje de las redes neuronales.[19]

|  |  |
| --- | --- |
| Tipos de aprendizaje | Descripción |
| Aprendizaje Adaptativo | La capacidad de aprendizaje adaptativo es una de las características más atractivas de redes neuronales. Esto es, aprenden a llevar a cabo ciertas tareas mediante un entrenamiento con ejemplos ilustrativos. |
| Aprendizaje Supervisado | Estos algoritmos cuentan con un aprendizaje previo basado en un sistema de etiquetas asociadas a unos datos que les permiten tomar decisiones o hacer predicciones. Un ejemplo es un detector de spam que etiqueta un e-mail como spam o no dependiendo de los patrones que ha aprendido del histórico de correos (remitente, relación texto/imágenes, palabras clave en el asunto, etc.) |
| Aprendizaje No supervisado | Estos algoritmos no cuentan con un conocimiento previo. Se enfrentan al caos de datos con el objetivo de encontrar patrones que permitan organizarlos de alguna manera. Por ejemplo, en el campo del marketing se utilizan para extraer patrones de datos masivos provenientes de las redes sociales y crear campañas de publicidad altamente segmentadas. |
| Aprendizaje por refuerzo | Su objetivo es que un algoritmo aprenda a partir de la propia experiencia. Esto es, que sea capaz de tomar la mejor decisión ante diferentes situaciones de acuerdo a un proceso de prueba y error en el que se recompensan las decisiones correctas. En la actualidad se está utilizando para posibilitar el reconocimiento facial, hacer diagnósticos médicos o clasificar secuencias de ADN. |

Las redes neuronales y la inteligencia artificial en la actualidad han tenido grandes avances tecnológicos, pudiendo ser utilizada de diversas maneras dependiendo el propósito que se le quiera asignar, ya que aún no se logra una singularidad en este tipo de materia, por lo cual comúnmente crean inteligencias que se especialicen en un sector, y es por ello que la inteligencia artificial se puede dividir en 3 ramas principales.

**2.4.2 Rama de interés de la inteligencia artificial**

Todas las ramas de la inteligencia artificial están dedicadas a ser útiles en diversos tipos de sectores, como lo son el Machine Learning y la robótica con IA, no obstante, el principal referente en esta investigación es sobre la rama de los asistentes virtuales, los cuales son bastante populares en la actualidad.

**Asistentes Virtuales**

Son herramientas inteligentes que ayudan a los usuarios a buscar información en un conglomerado de recursos web. El despliegue natural de los mismos se realiza en las propias páginas web, donde permiten resolver las dudas de los usuarios formuladas en lenguaje natural usando técnicas de Inteligencia Artificial.

En particular, un Agente Virtual o chatbot puede ser utilizado en el ámbito educativo como Asistente Virtual de Clase, AVC, a modo de complemento en el proceso enseñanza aprendizaje. De forma tal que, con determinados contenidos y rutinas específicas, pueda interactuar con los alumnos, brindar información, realizar tutorías, tomar exámenes, etc. La interacción en lenguaje natural es posible gracias al empleo de una rama de Inteligencia Artificial conocida como PLN (Procesamiento del Lenguaje Natural), que trata sobre cómo las máquinas pueden comprender el lenguaje del hombre. [19]

Los asistentes virtuales son de gran utilidad y facilitan diversas tareas al usuario, pudiendo realizarse la comunicación a través de la inserción de texto o a través del lenguaje natural, siendo que para este caso el asistente debe tener la capacidad de realizar el procesamiento de lenguaje natural, para tener una comunicación fluida entre el usuario y máquina.

## 2.5 Procesamiento de Lenguaje Natural

El procesamiento del lenguaje natural consiste en la habilidad de una máquina para procesar información comunicada mediante el uso del lenguaje natural. Crean modelos computacionales del lenguaje suficientemente detallados que permitan escribir programas informativos que realicen distintas órdenes o peticiones donde interviene el lenguaje natural. Se podría decir que el NLP consiste en usar una expresión natural que pueda tener comunicación con la computadora directamente, por medio escrito o comando de voz, facilitando las órdenes o peticiones con el lenguaje, o seguir desarrollando modelos que ayuden a la comprensión humana y sus mecanismos que se relacionan al lenguaje. [19]

El procesamiento de lenguaje natural es de gran importancia en los asistentes virtuales, ya que esto permite que la maquina entienda al usuario, sea a través de la escritura de la petición por medio del teclado, o por medio del habla del usuario, y es por ello que el asistente debe tener integrado un módulo de reconocimiento de voz para poder comprender al usuario.

### 2.5.1 Reconocimiento de Voz

El reconocimiento del habla permite a un ser humano comunicarse con un ordenador. A grandes rasgos, consiste en que el ordenador captura la señal de voz que emite una persona a través de un micrófono, convirtiéndola en información digital. El motor de voz debe ser capaz de reconocer las silabas de entre un conjunto de fonemas que ha recibido, y combinarlas para formar las palabras que se habían dicho anteriormente por el usuario. [21].

El reconocimiento de voz básicamente funciona en 3 etapas, primero se ingresa el audio sea a través de un micrófono o un archivo de texto, la siguiente etapa es hacer el reconocimiento y al final pasar las palabras reconocidas a un archivo de texto, esto se puede ver en la Figura 1.



Figura 1 Etapas del reconocimiento de voz

Aunque las fases para el reconocimiento de voz pueden sonar muy simples, el proceso de cómputo que se realiza es algo elevado. Existe una clasificación de técnicas en las que se puede implementar este tipo de tecnologías, pudiendo ser útiles para el contexto del software que se va a desarrollar, estas se observan en la tabla 2.

Tabla 2 Tipos de reconocimiento de voz. [21]

|  |  |
| --- | --- |
| Reconocimiento | Descripción |
| Reconocimiento Automático del Locutor (RAL) | Un sistema de reconocimiento automático del locutor permite al sistema comprobar si la persona que ha emitido la señal de voz, es en verdad quien dice ser. Para ello hay realizar un entrenamiento al sistema. El locutor debe introducir muestras de voz para que el sistema pueda crear una serie de patrones. Una vez que se ha hecho esto, se dice que el sistema esta entrenado y está listo para reconocer al locutor. |
| Reconocimiento Automático del Habla (RAH) | El reconocimiento automático del habla presenta varias modalidades producto de una serie de restricciones que se le imponen a la tarea de reconocimiento con el fin de simplificarla. Los principales parámetros que se utilizan para realizar este tipo de reconocimiento son los siguientes: modalidad de habla, estilo de habla, entrenamiento, tamaño del vocabulario, modelo del lenguaje, etc. Un sistema de Reconocimiento Automático del Habla ideal, es aquel que funciona en entornos con ruido de fondo muy alto, y es capaz de reconocer el habla de cualquier locutor, corrige los errores producidos por la mala pronunciación de éste y además es insensible a las variaciones inducidas por los canales de comunicación, pero aún no se ha conseguido este sistema |

El reconocimiento de voz es primordial si el asistente tiene el objetivo de mantener una comunicación con el usuario lo más natural posible, sin embargo, el hecho de reconocer la voz es una parte, la siguiente parte es como el asistente va a responder al usuario, ya sea de manera textual o por medio del habla, este último se le conoce como síntesis de voz, el cual permite una comunicación más natural entre el hombre y la máquina.

### 2.5.2 Síntesis de Voz

La Síntesis de Voz, también conocida como Conversión de Texto a Voz (CTV), consiste en dotar al sistema de la capacidad de convertir un texto dado en voz. Esto se puede hacer mediante grabaciones realizadas anteriormente por personas. La voz del ordenador puede generarse uniendo las grabaciones que se han hecho, ya sean de palabras enteras, o fonemas, pero siempre intentando que el sonido producido parezca lo más natural posible e inteligible, encadenando correctamente los sonidos dentro del discurso. [22]

Los tipos de síntesis de voz se pueden clasificar en dos, y se basan en cómo es que generan y reproducen el archivo de audio para comunicarse. Los tipos de síntesis de voz se pueden ver en la tabla 3. Si el objetivo de este proyecto fuera desarrollar un motor de reconocimiento y síntesis de voz, la solución más sencilla seria realizar la síntesis Concatenativa ya que únicamente se tienen que concatenar los registros de audio para generar palabras, sin embargo, como este no es el objetivo general, se utilizara una API abierta, y estas normalmente hacen uso de la síntesis acústica para una mayor claridad por parte del asistente.

Tabla 3 Tipos de síntesis de voz [22]

|  |  |
| --- | --- |
| Tipos de Síntesis de voz | Descripción |
| Síntesis Concatenativa | La Síntesis Concatenativa se basa en la unión de segmentos de voz grabados. Este método produce una síntesis más natural, pero se pierde a causa de las variaciones del habla. Existen tres métodos para realizar la Síntesis Concatenativa. La llamada Síntesis por selección de unidades utiliza una base de datos en la que se encuentran grabaciones de voz tanto de fonemas, silabas, palabras, frases y oraciones. Este método es el que produce un sonido más natural, pero estas bases de datos pueden alcanzar un tamaño muy grande. |
| Síntesis de formantes | Este método no utiliza muestras de habla humana en tiempo de ejecución como los anteriores, sino que se utiliza un modelo acústico. Se crea una onda de habla artificial. Este método produce un sonido robótico y nunca se podría confundir con la voz humana, pero tiene la ventaja de que producen programas más pequeños ya que no necesitan de una base de datos de muestras grabadas como los métodos de concatenación. |

### 2.5.3 Softwares útiles para el procesamiento de lenguaje natural.

El procesamiento del lenguaje natural es la tecnología sobre la cual muchos expertos han puesto su enfoque, ya que la comunicación entre el hombre y la máquina es algo bastante importante para que un sistema computacional pueda procesar las tareas asignadas de una manera correcta. Si una máquina no entiende las instrucciones que el usuario quiere que realice, la computadora no realizara ninguna acción o la implementara incorrectamente.

En la actualidad, existen diversidad de herramientas que nos permiten implementar el procesamiento del lenguaje natural para que una computadora comprenda al usuario; estas herramientas pueden usarse a través de framework, API’s, complementos Web, etc. Algunos son de licencia privativa y otras sin de uso libre. Considerar cada una de las alternativas es de gran importancia para el proyecto, estas se pueden ver en la tabla 4.

Cada una de las herramientas para el reconocimiento de voz tienen ventajas y desventajas de funcionamiento, en este caso la herramienta elegida para el reconocimiento principal es Speech Recognition, desarrollada como una API por Google.

Tabla 4 Herramientas para el procesamiento del lenguaje natural [23]

|  |  |  |
| --- | --- | --- |
| Herramienta | Definición | Características |
| KALDI | Tipo especial de software de reconocimiento de voz, iniciado como parte de un proyecto de la Universidad John Hopkins. Este kit de herramientas viene con un diseño extensible y escrito en el lenguaje de programación C++. | Una aplicación de reconocimiento de voz de código abierto, libre y flexible, bajo la licencia Apache.  Proporciona soporte para instalar y configurar la aplicación en su sistema.  Además del sistema de reconocimiento de voz, también soporta redes neuronales profundas y transformaciones lineales. |
| CMUSphinx | CMUS Sphinx viene con un grupo de sistemas enriquecidos con varias funciones con diferentes paquetes prediseñados relacionados con el reconocimiento de voz. Es un programa de código abierto, desarrollado en la Universidad Carnegie Mellon. | Viene con un diseño flexible y un sistema eficiente.  Proporciona herramientas de entrenamiento de modelos acústicos a través de su paquete Sphinxtrain.  Ayuda a realizar diferentes tipos de tareas a través de sus útiles paquetes, incluyendo la detección de palabras clave, la evaluación de la pronunciación, la alineación, y más. |
| DeepSpeach | DeepSpeech es un motor de reconocimiento de voz de código abierto para convertir su voz en texto. Es una aplicación gratuita de Mozilla. Para ejecutar el proyecto DeepSearch, se necesita Python 3.r o superior. | DeepSpeech utiliza el marco de trabajo TensorFlow  Es compatible con la GPU NVIDIA.  Puede utilizar la inferencia de DeepSearch de tres maneras diferentes: El paquete Python, el paquete Node.JS o el cliente de línea de comandos. |
| Mycroft | Mycroft viene con un asistente de voz de código abierto fácil de usar para convertir la voz en texto. Está considerado como una de las herramientas de reconocimiento de voz de Linux más populares en la era moderna, escrita en Python. | Integrado con los medios sociales y plataformas profesionales más populares, incluyendo Facebook, Github, LinkedIn, y más.  Puede ejecutar esta aplicación en diferentes plataformas de software y hardware.  Además de ser un asistente de voz inteligente, proporciona la facilidad de la grabación de audio, aprendizaje automático, biblioteca de software, y más |

# III. Metodología

La metodología que se implementara para llevar a cabo este proyecto se basa en la metodología ágil de software, SCRUM, debido a que el campo de desarrollo de este proyecto requiere mucha programación, esta metodología es idónea, teniendo en cuenta su particularidad de implementación de un desarrollo dividido por “Sprint”, que equivale a una cierta cantidad de tiempo en el que plantea hacer la implementación de aspectos específicos de la lista de requisitos generales, acelerando el tiempo de desarrollo del software. Las fases y flujo de trabajo de la metodología se pueden ver en la Figura 2.

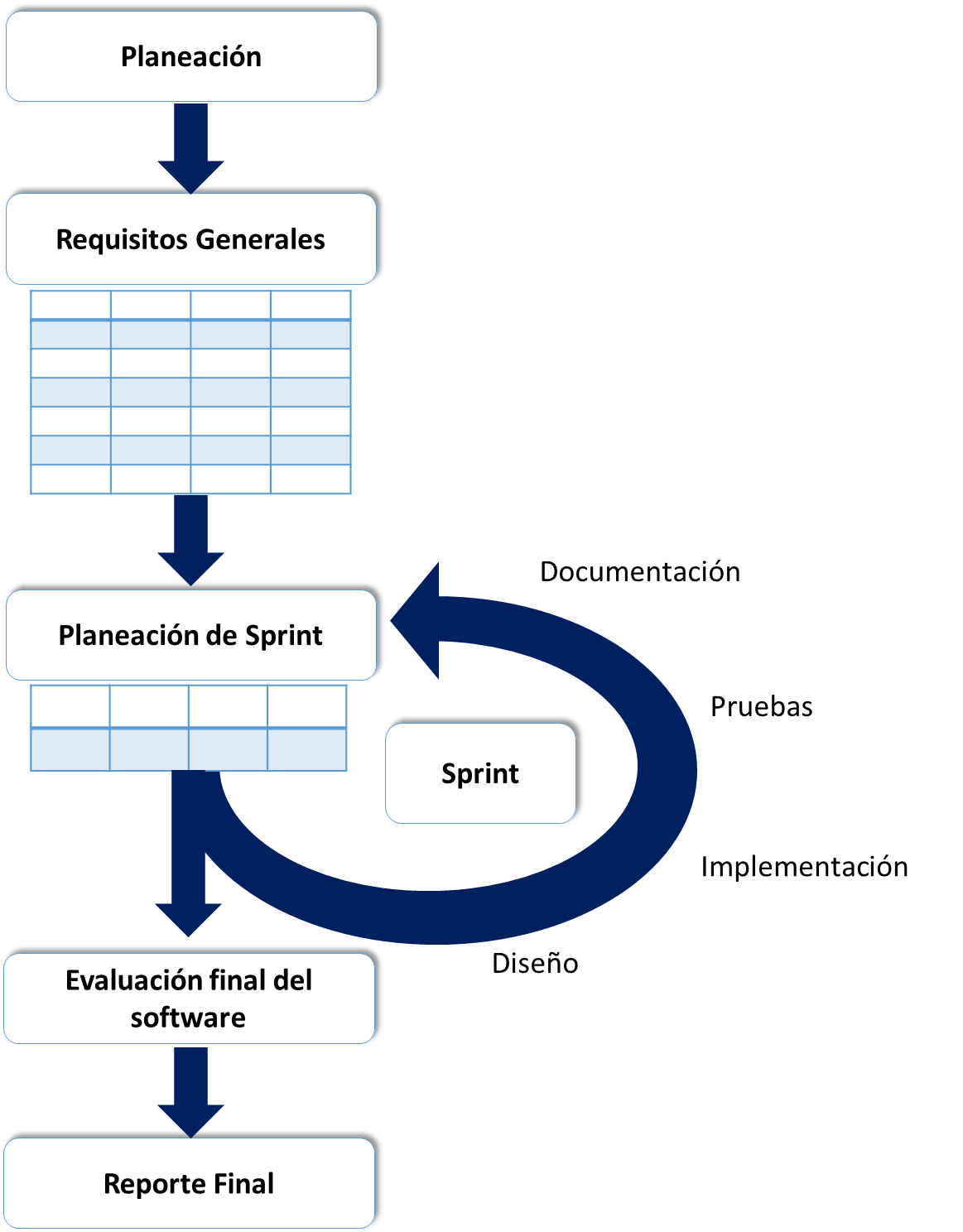


Figura 2 Diagrama metodología basada en SCRUM

En cada una de las fases de la metodología se pretenden desarrollar ciertas actividades en específico, a continuación, se presenta una descripción general y en qué consiste cada fase de la misma:

1. **Planificación**

En la etapa de planificación se desarrollarán diversas actividades que fungirán como guía para el inicio y culminación del proyecto, definiendo el tiempo inicio del desarrollo y la fecha tentativa para finalizarlo, las tareas que se realizarán de manera general y por quien serán realizadas por parte del equipo de trabajo. Esto a través del uso de esquemas que permitirán mantener el proyecto en dirección del cumplimiento de los objetivos.

1. **Requisitos Generales**

Los requisitos del software es un aspecto fundamental en esta metodología, ya que con base al producto que da como resultado esta fase, se podrá realizar la planificación de cada uno de los Sprint. Durante esta etapa se definen los requisitos funcionales y no funcionales del software, el hardware necesario para utilizarlo, las características lógicas y visuales del software, así como los requisitos de usuario llamados casos de uso en donde se definirán cada uno de los usos que el software puede tener.

1. **Planeación de Sprint**

Se definen los objetivos que se plantean lograr alcanzar en el sprint, conformando esto una parte de los requisitos generales recopilados en la fase anterior de requisitos generales, asignando a su vez que tareas realizará cada miembro del equipo de trabajo y determinando el tiempo en el que se pretende realizar ese sprint.

1. **Sprint**

El Sprint es la etapa donde se realiza la implementación del proyecto, en el cual se realizan actividades importantes que incrementan el avance general del proyecto. El sprint cuenta con diversas sub-Fases, las cuales se pueden observar en la Figura 3, que se realizan de manera cíclica junto a la fase de Planeación de Sprint hasta que el desarrollo finalice.

* **Diseño:** En esta etapa se diseña el funcionamiento lógico del software determinando como será el flujo de funcionamiento, el diseño técnico que define las tecnologías que se utilizan y el diseño visual a través de diversos prototipos que más adelante se programaran.
* **Implementación:** Esta fase es donde se realiza la codificación del software, en la cual, con base al diseño, se utilizan las tecnologías elegidas y se implementan los diversos algoritmos necesarios para que el software funcione como lo esperado.
* **Pruebas:** Durante las Pruebas, se hace una evaluación de tipo unitaria de lo desarrollado en el Sprint, asegurándose de que lo implementado funcione correctamente y en caso de que existan detalles se contemplan como corrección en el siguiente Sprint.
* **Documentación:** Esta etapa es el final del Sprint, en el cual se crea un documento donde se detalla lo realizado durante este lapso de tiempo en cuestión de las fases previas.

Figura 3 Fases dentro del Sprint

1. **Evaluación Final del software**

Ejecución de una serie de pruebas rigurosas al software con el fin de determinar si el objetivo planteado en la fase de planeación se logró alcanzar, obteniendo estadísticas con base a cada uno de las pruebas.

1. **Reporte Final**

Se realiza un reporte general del proyecto realizado, redactando todas las funcionabilidades del producto desarrollado, el proceso seguido desde su inicio y un análisis de los resultados obtenidos, para que al final el proyecto se dé por concluido.

**3.1 Planificación**

El presente proyecto se planteó desarrollarse en un tiempo de 2 meses, donde la primera parte se contempla que sea el diseño y codificación del mismo, por lo que es la que hará uso de más tiempo a partir de la fecha 25 de abril al 23 de mayo del 2022, y la segunda parte será la evaluación del software para determinar si el proyecto realizado cumplió con los objetivos planteados al inicio del proyecto. En la figura se puede 4 se puede ver el cronograma de actividades realizado con base a los objetivos de este proyecto. 

Figura 4 Cronograma de Actividades

El tiempo es un factor de gran importancia al realizar cualquier tipo de proyecto, y es por ello que se distribuyeron las actividades de tal forma que terminen correctamente.

**Actividades**

* Se realizará la investigación de los antecedentes de los asistentes virtuales enfocados a la educación, la actividad la realizará Ángel y Julio.

(Búsqueda de información sobre los antecedentes de asistentes virtuales enfocados a la educación, en páginas web, revistas, artículos, tesis, trabajos de investigación)

(Intercambio de información obtenida por Ángel y Julio de distintas fuentes)

* Se investigarán los requerimientos de Sistema y sistemas de conversión de texto a voz enfocados a la educación, actividad a realizar por Ángel y Julio.

(Recopilación de información sobre los requerimientos de sistema y sistemas de conversión de texto a voz enfocados a la educación, en revistas, artículos, páginas web, trabajos de investigación, informes, documentación)

* Diseñar la estructura del sistema.

(Creación del diseño para la estructura del sistema, diseño de interfaces que sean amigables con el usuario, de bajos recursos y con un diseño minimalista)

* Definir un formato de almacenamiento de código de manera local

(Definir un formato de almacenamiento de código mediante ficheros, un identificador de código, largo de código, código y un identificador de salida)

* Implementar el motor de voz a texto

(De acuerdo con las búsquedas relacionadas, elegir el sistema de conversión ya implementado de código abierto, dando prioridad a los motores realizados en Python).

* Implementar un sistema de identificación de palabras clave

(Eliminar los conectores de las palabras, dejándolas en crudo, así como las más comunes al iniciar un buscador como: el nombre del asistente y palabras como “*busca, muéstrame u obtén*”).

* Crear el módulo para la búsqueda de información

(Se implementará un módulo basado en web scraping con Python con target en los diferentes repositorios que se hayan seleccionado).

* Evaluar la funcionalidad del asistente

(Realizar diferentes pruebas al asistente, tanto unitarias como de estrés)

* Documentar el desarrollo y resultados del proyecto

(A lo largo del proyecto se documentará todo el proceso y las acciones tomadas con el asistente virtual).

## 3.2 Requisitos del software

### 3.2.1 Requisitos funcionales

Los requerimientos funcionales son aquellos que se refieren a los servicios que el sistema puede realizar con base a determinadas acciones. A través de algún tipo de entrada que se realice sobre el software, este generará algún tipo de respuesta que recibirá el usuario que está manipulando el sistema. En el caso del presente proyecto, los requisitos funcionales determinados se detallan en la tabla 5.

Tabla 5 Requisitos funcionales

|  |  |  |
| --- | --- | --- |
| **ID** | **Nombre del requerimiento** | **Descripción** |
| **RF01** | Interfaz minimalista | El sistema debe contar con una interfaz minimalista y fácil de entender. |
| **RF02** | Modalidad de ventanas Flotantes | El sistema debe contar con el tipo de visualización a través de ventanas flotantes. |
| **RF03** | Modalidad de ventana extendida | El sistema debe contar con la modalidad de ventana extendida, con el mismo tipo de funcionalidad que con ventanas flotantes. |
| **RF04** | Solicitud por entrada de voz | El sistema puede reconocer la voz y entender la solicitud del usuario. |
| **RF05** | Solicitud por medio del teclado | El sistema puede recibir el texto ingresado por el teclado y entender la solicitud del usuario. |
| **RF06** | Identificación de palabras clave. | Al recibir la solicitud del usuario, se debe de identificar con base a palabras clave que es lo que necesita. |
| **RF07** | Repositorio local para el manejo de búsquedas | Se debe contar con un archivo de los sitios seleccionados para realizar la búsqueda web. |
| **RF-08** | Búsqueda Web | A través de la solicitud el sistema realiza la búsqueda en la web para encontrar coincidencias (Web Scraping) |
| **RF-09** | Archivo local de búsqueda | El sistema debe contar con un archivo almacenado de manera local para almacenar bloques de código y realizar búsquedas. |
| **RF-10** | Búsqueda Local | El sistema puede realizar búsquedas en el archivo local con base al identificador de código. |
| **RF-11** | Registro de bloque de código | Con base a un identificador y el bloque de código se pueden almacenar en el archivo local. |
| **RF-12** | Síntesis de voz | A través de la conversión de texto a voz, el sistema responderá aspectos específicos al usuario. |
| **RF-13** | Configuración de interfaz | El usuario puede configurar aspectos de la interfaz para una mayor comodidad. |
| **RF-14** | Configuración de síntesis de voz | En el apartado configuración se pueden retocar aspectos de funcionamiento del sintetizador de la voz. |

### 3.2.2 Requisitos no funcionales

Los requisitos no funcionales a diferencia de los funcionales no representan algún servicio o función que tiene el software, si no que estos hacen referencia a propiedades del sistema como lo son relacionados al rendimiento, a la seguridad, la portabilidad, usabilidad etc. En el caso del asistente que se está desarrollando, los requisitos no funcionales definidos se pueden ver en la tabla 6.

Tabla 6 Requisitos no funcionales

|  |  |  |
| --- | --- | --- |
| **ID** | **Nombre del requerimiento** | **Descripción** |
| **RNF-01** | Navegabilidad | El sistema deberá contar con una interfaz sencilla e intuitiva para el usuario. |
| **RNF-02** | Validación de datos | El sistema no debe de permitir realizar acciones cuando los campos de texto no cumplen con los requisitos adecuados. |
| **RNF-03** | Control de errores | El sistema debe de mantener una funcionalidad correcta sin importar de que surjan errores. |
| **RNF-04** | Accesibilidad a archivos de configuración | El sistema puede ser configurado directamente en sus archivos de configuración. |
| **RNF-05** | Disponibilidad sin conexión | En caso de que no exista conexión a la red, el sistema debe seguir funcionando para solicitudes locales. |
| **RNF-06** | Ayuda para el manejo del sistema | El sistema debe contar con la opción de orientar al usuario de cómo funciona el sistema. |
| **RNF-07** | Desempeño | El rendimiento el sistema debe de mantenerse fluido, respondiendo a las solicitudes del usuario rápidamente. |
| **RNF-08** | Aplicación de escritorio | La aplicación deberá contar la característica de ser de tipo escritorio, instalándose dentro de un sistema y funcionando al instante de su ejecución. |

### 3.2.3 Diagrama de casos de uso

Para el desarrollar cualquier tipo de proyecto, es importante determinar cuál va a ser el uso que este va a tener cuando se haya completado, sin embargo, dentro de un software no solo se puede presentar un solo tipo de uso, ya que entre las interfaces quizá se puedan realizar diversos tipos de acciones con las que el usuario puede interaccionar y es por ello que determinar los caso de uso antes del desarrollo es un requisito importante ya que con base a ese modelo se pueden determinar cada uno de los usos que va a tener el software y de esta manera se pueda desarrollar tal cual se planeó.

En el caso de este asistente basado en el procesamiento del lenguaje natural, se tienen diversos casos de uso puntuales, los cuales en la figura 4 se pueden observar de manera clara, en la cual tan solo un actor dentro del sistema, el usuario, quien es el que va a hacer uso del asistente y mandar las diversas solicitudes al software para después recibir un tipo de respuesta. Entre los casos de uso que este software tiene planeado implementar es que el usuario pueda iniciar la aplicación, hacer registros locales para posibles consultas futuras, realizar solicitudes de voz o por medio del teclado a través de la web o de manera local, recibir respuesta de su solicitud a través de la interfaz o hacer diversos cambios en la configuración de la aplicación.

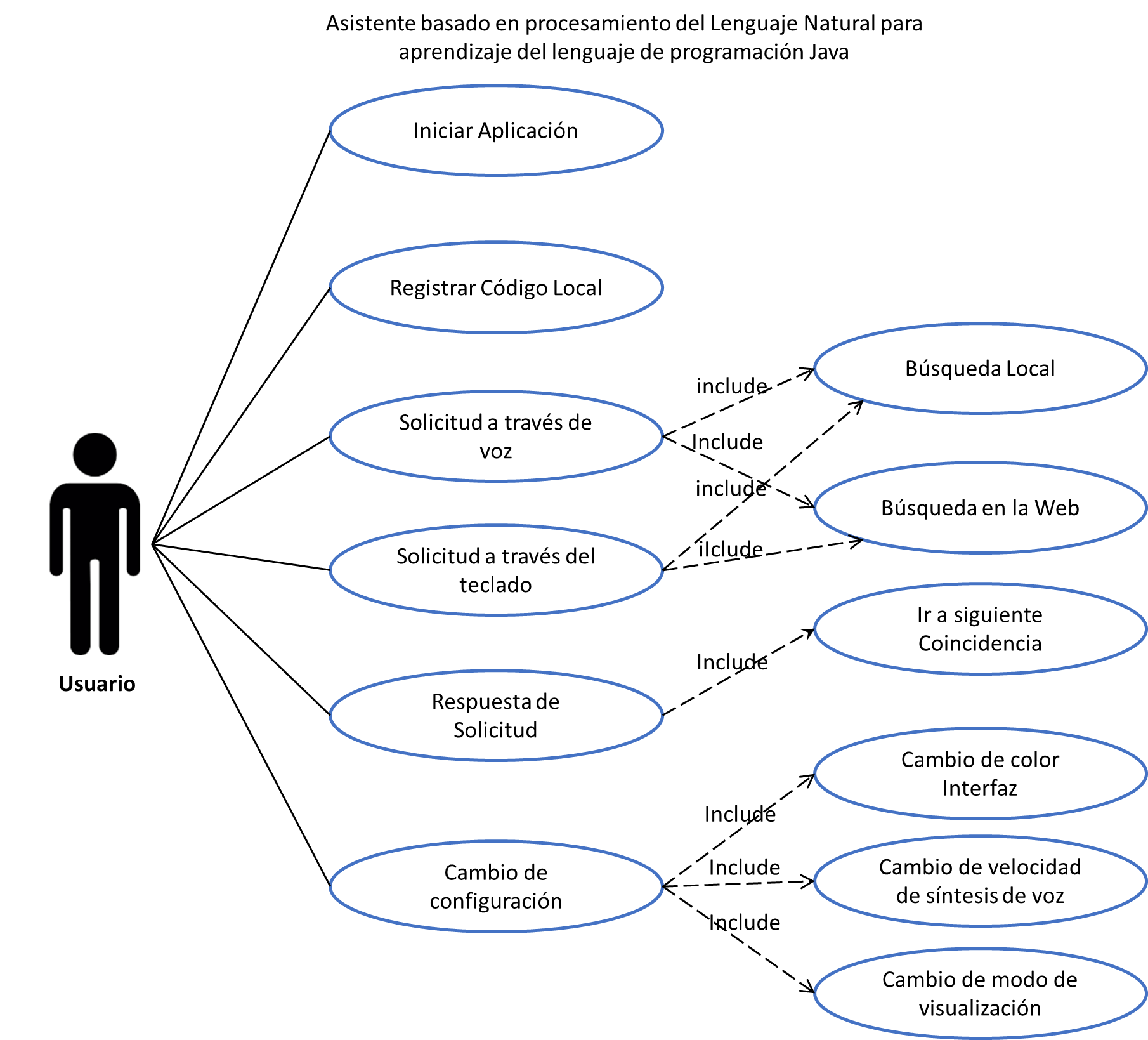


Figura 5 Diagrama de casos de uso

### 3.2.4 Actores en el sistema

El software a desarrollar únicamente contempla que solo un tipo de actor va a hacer uso del sistema, y en este caso es un usuario, el cual puede acceder a todas las acciones que pueden hacerse en el sistema, las cuales están representadas en la figura 4. En la tabla 5, en la cual se describe el tipo de actor que es y cada uno de los casos de uso en los que está involucrado.

Tabla 7 Tabla de actores del sistema: Usuario

|  |  |
| --- | --- |
| **Actor** | Usuario |
| **Tipo** | Primario |
| **Casos de uso** | Iniciar aplicación, Registrar código local, solicitud a través de voz, solicitud a través del teclado, búsqueda local, búsqueda en la web, respuesta de solicitud, ir a siguiente coincidencia, cambio de configuración, cambio de color de interfaz, cambio de velocidad de síntesis de voz, cambio de modo de visualización. |

### 3.2.5 Casos de uso

En el caso de este software a desarrollar, se abarcan diversos aspectos que se pueden considerar casos de uso, como lo es realizar una solicitud, obtener información o bien configurar parámetros dentro del software. Los casos de uso se presentan de manera más detallada en la tabla 8 a la tabla 19.

Tabla 8 Caso de uso: Iniciar aplicación

|  |  |
| --- | --- |
| **Caso de uso** | Iniciar aplicación |
| **Identificador** | 1 |
| **Actores** | Usuario |
| **Tipo** | Básico |
| **Propósito** | Comenzar con la ejecución del programa. |
| **Resumen** | Es un caso de uso básico pero esencial, ya que esto inicia la interfaz del software. |
| **Precondiciones** | Debe existir disponibilidad de recursos dentro del dispositivo. |
| **Secuencia normal** | 1. Buscar aplicación. 2. Doble clic sobre el icono. |
| **Post- condiciones** | La interfaz del software se muestra en pantalla. |
| **Excepciones** | E1. Falta de recursos para abrir la aplicación. |

Tabla 9 Caso de uso: Registrar Código Local

|  |  |
| --- | --- |
| **Caso de uso** | Registrar Código Local |
| **Identificador** | 2 |
| **Actores** | Usuario |
| **Tipo** | Básico |
| **Propósito** | Realizar la inserción de bloques de código de manera local. |
| **Resumen** | Realiza la inserción de código con base a un identificador y el bloque de código que se almacenara dentro de un fichero local. |
| **Precondiciones** | El archivo local debe estar creado. |
| **Secuencia normal** | 1. Abrir aplicación 2. Dar clic sobre el icono para el código local 3. Ingresar identificador. 4. Ingresar bloque de código. 5. Dar clic en ingresar |
| **Post- condiciones** | El asistente responde si el proceso se realizó exitosamente o si existió algún tipo de error. |
| **Excepciones** | E1. El identificador de código coincide con alguno otro previamente registrado.  E2. Falta ingresar el identificador de código.  E3. Alguno de los campos de código se encuentra vacíos. |

Tabla 10 Caso de uso: Solicitud a través de voz

|  |  |
| --- | --- |
| **Caso de uso** | Solicitud a través de voz |
| **Identificador** | 3 |
| **Actores** | Usuario |
| **Tipo** | Básico |
| **Propósito** | A través del micrófono, el usuario puede realizar algún tipo de solicitud al asistente. |
| **Resumen** | El usuario puede realizar una solicitud local o a través de la web con el uso de su voz y el micrófono. |
| **Precondiciones** | El micrófono debe estar disponible y debe existir poco ruido en el ambiente |
| **Secuencia normal** | 1. Abrir aplicación 2. Dar clic sobre el icono de micrófono 3. Pulsar el icono para entrada de voz. 4. Hablar la solicitud |
| **Post- condiciones** | El sistema crea un frame con el resultado de la búsqueda. |
| **Excepciones** | E1. El micrófono está ocupado.  E2. No se entendió la solicitud del usuario. |

Tabla 11 Caso de uso: Solicitud a través del teclado

|  |  |
| --- | --- |
| **Caso de uso** | Solicitud a través de teclado |
| **Identificador** | 4 |
| **Actores** | Usuario |
| **Tipo** | Básico |
| **Propósito** | A través del teclado, el usuario puede realizar algún tipo de solicitud al asistente. |
| **Resumen** | El usuario puede realizar una solicitud local o a través de la web con da la escritura por el teclado. |
| **Precondiciones** | La interfaz debe de estar abierta. |
| **Secuencia normal** | 1. Abrir aplicación 2. Dar clic sobre el icono de teclado 3. Escribir la solicitud para el asistente. 4. Pulsar el icono de enviar |
| **Post- condiciones** | El sistema crea un frame con el resultado de la búsqueda. |
| **Excepciones** | E1. No se entendió la solicitud del usuario. |

Tabla 12 Caso de uso: Búsqueda local

|  |  |
| --- | --- |
| **Caso de uso** | Búsqueda Local |
| **Identificador** | 5 |
| **Actores** | Usuario |
| **Tipo** | Básico |
| **Propósito** | Realizar una búsqueda en el fichero local |
| **Resumen** | Se realiza la búsqueda de identificado de código dentro del archivo local. |
| **Precondiciones** | Debe existir una solicitud por medio de la voz o el teclado por parte del usuario. |
| **Secuencia normal** | 1. Solicitud a través de la voz o teclado. 2. Se detecta que en la solicitud esta la frase local. |
| **Post- condiciones** | El sistema crea un frame con el resultado de la búsqueda local. |
| **Excepciones** | E1. No se encontró el resultado.  E2. El archivo local esta dañado.  E3. El archivo local no se encuentra. |

Tabla 13 Caso de uso: Búsqueda en la web

|  |  |
| --- | --- |
| **Caso de uso** | Búsqueda en la Web |
| **Identificador** | 6 |
| **Actores** | Usuario |
| **Tipo** | Básico |
| **Propósito** | Realizar una búsqueda en la web. |
| **Resumen** | Se realiza la búsqueda de identificado de código dentro de los sitios web del archivo de repositorios. |
| **Precondiciones** | Debe existir una solicitud por medio de la voz o el teclado por parte del usuario. |
| **Secuencia normal** | 1. Solicitud a través de la voz o teclado. 2. Se detectan las palabras claves. 3. Se consulta el archivo de repositorios web. 4. Se realiza la búsqueda en el primer sitio, si no existe se dirige al otro y así secuencialmente hasta encontrarlo |
| **Post- condiciones** | El sistema crea un frame con el resultado de la búsqueda. |
| **Excepciones** | E1. No se encontró el resultado.  E2. No existe conexión a la red. |

Tabla 14 Caso de uso: Respuesta de solicitud

|  |  |
| --- | --- |
| **Caso de uso** | Respuesta de solicitud |
| **Identificador** | 7 |
| **Actores** | Usuario |
| **Tipo** | Básico |
| **Propósito** | El sistema devuelve una respuesta al usuario. |
| **Resumen** | A partir de una petición del usuario, el sistema realiza algún tipo de búsqueda y retorna el resultado en un frame al usuario. |
| **Precondiciones** | El usuario debió realizar algún tipo de solicitud. |
| **Secuencia normal** | 1. Se da clic en enviar solicitud. 2. El sistema realiza la búsqueda. 3. Se crea un frame con el resultado de la solicitud. |
| **Post- condiciones** | Se crea un frame con la respuesta a la solicitud y el asistente le avisa al usuario. |
| **Excepciones** | Sin excepciones. |

Tabla 15 Caso de uso: Ir a siguiente coincidencia

|  |  |
| --- | --- |
| **Caso de uso** | Ir a siguiente coincidencia |
| **Identificador** | 8 |
| **Actores** | Usuario |
| **Tipo** | Básico |
| **Propósito** | Ir a otro sitio para ver otra alternativa de respuesta a la solicitud. |
| **Resumen** | Al desplegar el resultado de la búsqueda, el usuario puede dar clic a siguiente resultado en caso de que no le convenza el actual. |
| **Precondiciones** | Se tuvo que realizar alguna búsqueda web. |
| **Secuencia normal** | 1. Se realiza la búsqueda. 2. Se muestra el frame con el resultado. 3. Se da clic en siguiente resultado. |
| **Post- condiciones** | Se muestra el nuevo frame con el nuevo resultado encontrado. |
| **Excepciones** | E1. No hay más resultados.  E2. Sin conexión a internet. |

Tabla 16 Caso de uso: Cambio de configuración

|  |  |
| --- | --- |
| **Caso de uso** | Cambio de configuración |
| **Identificador** | 9 |
| **Actores** | Usuario |
| **Tipo** | Básico |
| **Propósito** | Realizar una modificación general del sistema |
| **Resumen** | El usuario puede modificar diversos atributos dentro del software. |
| **Precondiciones** | La interfaz debe de estar abierta |
| **Secuencia normal** | 1. Abrir la aplicación 2. Seleccionar icono de configuración |
| **Post- condiciones** | Se reiniciará la aplicación y se efectuaran los cambios en el funcionamiento del software |
| **Excepciones** | E1. Archivo de configuración dañado. |

Tabla 17 Caso de uso: Cambio de color de interfaz

|  |  |
| --- | --- |
| **Caso de uso** | Cambio de color interfaz |
| **Identificador** | 10 |
| **Actores** | Usuario |
| **Tipo** | Básico |
| **Propósito** | Cambiar la apariencia de la interfaz. |
| **Resumen** | El usuario puede elegir entre una interfaz obscura o clara. |
| **Precondiciones** | El frame de configuración debe de estar abierto. |
| **Secuencia normal** | 1. Dar clic en configuración 2. Dirigirse al apartado de tema. 3. Seleccionar entre tema claro u obscuro. |
| **Post- condiciones** | Se reiniciará la aplicación y se efectuaran los cambios en el funcionamiento del software |
| **Excepciones** | E1. Archivo de configuración dañado. |

Tabla 18 Caso de uso: Cambio de velocidad de síntesis de voz

|  |  |
| --- | --- |
| **Caso de uso** | Cambio de velocidad de síntesis de voz |
| **Identificador** | 11 |
| **Actores** | Usuario |
| **Tipo** | Básico |
| **Propósito** | Cambiar la velocidad de habla del asistente. |
| **Resumen** | El usuario puede elegir la velocidad con la que responde el asistente. |
| **Precondiciones** | El frame de configuración debe de estar abierto. |
| **Secuencia normal** | 1. Dar clic en configuración 2. Dirigirse al apartado de voz del asistente 3. Ingresar el valor numérico de velocidad. |
| **Post- condiciones** | Se reiniciará la aplicación y se efectuaran los cambios en el funcionamiento del software |
| **Excepciones** | E1. Archivo de configuración dañado. |

Tabla 19 Caso de uso: Cambio de modo de visualización

|  |  |
| --- | --- |
| **Caso de uso** | Cambio de modo de visualización |
| **Identificador** | 12 |
| **Actores** | Usuario |
| **Tipo** | Básico |
| **Propósito** | Cambiar el modo de visualización de la interfaz. |
| **Resumen** | El usuario puede elegir entre 3 modos de visualización, de manera flotante vertical y horizontal, así como modo pantalla completa. |
| **Precondiciones** | El frame de configuración debe de estar abierto. |
| **Secuencia normal** | 1. Dar clic en configuración 2. Dirigirse al apartado de visualización. 3. Seleccionar entre los 3 modos. |
| **Post- condiciones** | Se reiniciará la aplicación y se efectuaran los cambios en el funcionamiento del software |
| **Excepciones** | E1. Archivo de configuración dañado. |

## 3.3 Planeación Sprint

El desarrollo como previamente se mencionó en la etapa de planeación, se consideró realizarse en 2 etapas, la primera que es la creación del asistente y la segunda es para realizar la evaluación de funcionamiento del software, por ende, con base a la metodología, las actividades a realizar se dividieron en diversos Sprint, los cuales en este caso corresponden a una semana, en la cual se desarrollaran diversas actividades para cumplir con una parte determinada de los requisitos generales que se recopilaron en la fase anterior.

En la tabla 20 se pueden ver los Sprint planteados y el tiempo en el que se van a desarrollar, así como las actividades y la persona responsable de ejecutar cada una de ellas.

Tabla 20 Planeación de Sprints para desarrollar el proyecto

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sprint** | **Inicio** | **Termino** | **Actividades** | **Responsable** |
| 1 | 25/04/2022 | 30/04/2022 | Determinar requisitos Funcionales | Julio Ponce |
| Determinar requisitos no funcionales | Ángel Maldonado |
| Generar diagrama de casos de uso | Julio Ponce |
| Detallar los casos de uso y su secuencia. | Ángel Maldonado |
| Documentación de Sprint | Julio Ponce y Ángel Maldonado |
| 2 | 02/05/2022 | 07/05/2022 | Diseño lógico del software | Julio Ponce |
| Prototipado de Interfaces | Ángel Maldonado |
| Codificación de interfaces flotantes | Julio Ponce |
| Implementación del manejador de ficheros | Ángel Maldonado |
| Pruebas Unitarias de la Interfaz y Controlador de ficheros | Julio Ponce y Ángel Maldonado |
| Documentación de Sprint | Julio Ponce y Ángel Maldonado |
| 3 | 09/05/2022 | 14/05/2022 | Diseño lógico para la implementación de reconocimiento / Síntesis de voz | Julio Ponce |
| Diseño del módulo de búsqueda local | Ángel Maldonado |
| Implementación del reconocimiento de voz | Julio Ponce |
| Desarrollo del módulo de identificación de palabras clave. | Julio Ponce y Ángel Maldonado |
| Pruebas Unitarias del reconocimiento de voz y búsqueda local. | Julio Ponce y Ángel Maldonado |
| Documentación de Sprint | Julio Ponce y Ángel Maldonado |
| 4 | 16/05/2022 | 21/05/2022 | Diseño del módulo de búsqueda web | Ángel Maldonado |
| Implementación del módulo de búsqueda Web | Julio Ponce y Ángel Maldonado |
| Desarrollo del apartado de configuración | Julio Ponce |
| Pruebas Unitarias del módulo de búsqueda Web | Julio Ponce y Ángel Maldonado |
| Documentación de Sprint | Julio Ponce y Ángel Maldonado |
| 5 | 23/05/2022 | 27/05/2022 | Optimización de módulos | Julio Ponce y Ángel Maldonado |
| Pruebas Unitarias | Julio Ponce y Ángel Maldonado |
| Documentación de Sprint | Julio Ponce y Ángel Maldonado |

## 3.4 Sprint

### 3.4.1 Diseño del software

**Diseño lógico del software**

Con base a los requisitos que se determinaron previamente en fases anteriores de la metodología, se definió de manera precisa cual es el funcionamiento lógico del asistente, el cual cuenta con diversos módulos que permiten que cumpla con el propósito para el que se programó. En la figura 6 se puede observar un diagrama que representa como como se ejecuta el asistente, teniendo diversos caminos a seguir dependiendo lo que el usuario decida realizar. En total son 4 módulos el dictado por voz para realizar una consulta sea de tipo web o local, la consulta a través del teclado que comparte instrucciones con el módulo previo; el módulo de guardar código de manera local, en el cual con base a un identificado y un bloque de código se puede almacenar en el archivo de códigos locales para consultas rápidas, así como el módulo de configuración, en el cual simplemente el usuario determina parámetros del software de su preferencia y los aplica. En cada acción que el usuario ejecuta el módulo de síntesis de voz (sea de forma online u offline) se encontrara activo, orientando al usuario a través de la conversión de texto a voz.

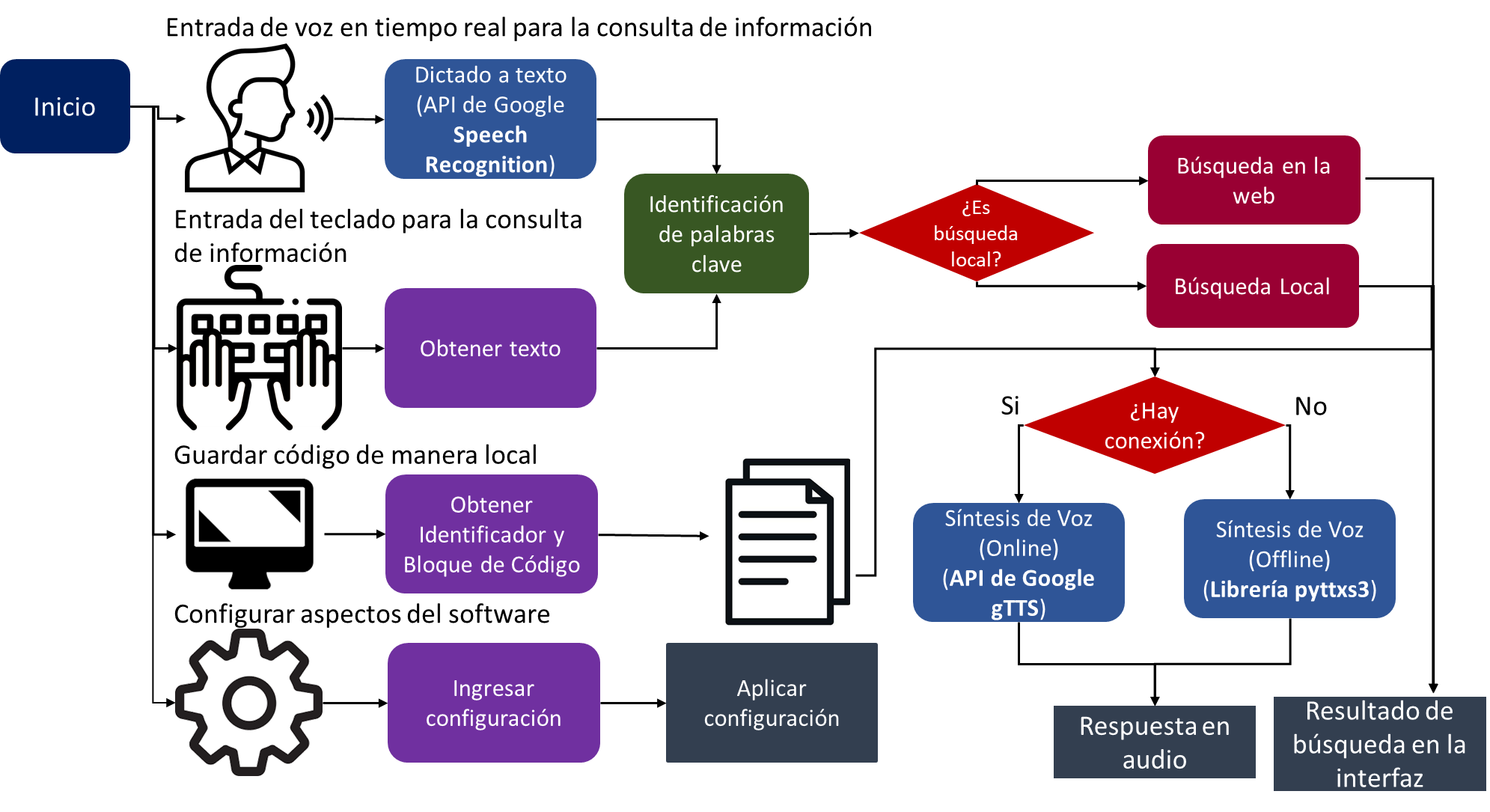


Figura 6 Diseño lógico del software

Las funcionalidades del software están estrechamente relacionadas a archivos que se almacenan localmente para facilitar el acceso a configuraciones o datos guardados por el usuario previamente, permitiendo que los datos que se requieran se consulten en el momento, evitando que se queden de manera dinámica en la memoria RAM del sistema y de esta manera manteniéndola libre para posibles tareas que el usuario desee realizar más adelante.

**Diseño de prototipos de Interfaz Grafica**

Un software comúnmente se desarrolla con una interfaz principal en la cual puedes acceder a todas sus funcionalidades sin la necesidad de salir de la ventana inicial, sin embargo, como en este proyecto se plantea el desarrollo de un asistente, la interfaz de este sería recomendable que sea como lo son los chatbots en los sitios web, los cuales se muestran con ventanas de tipo flotantes para que el usuario pueda acceder a estos en todo momento y no obstruya el contenido al que está accediendo. Con base a este principio es como se diseñará la interfaz gráfica del asistente, a través de ventanas flotantes.

El diseño general del software sigue un estilo minimalista, mostrando únicamente lo que el usuario necesita para hacer uso del software. En la figura 7 se muestra la distribución grafica de la ventana principal que se mostrara al ejecutar el software, en este caso se cuenta con 4 botones los cuales permiten acceder a cada uno de los módulos del asistente, como lo es entrada por voz y teclado, inserción de código local y configuración.

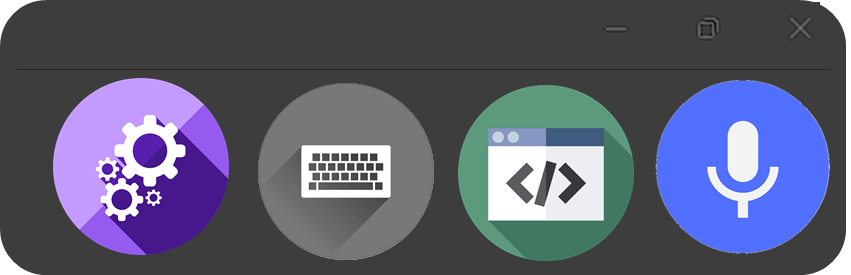


Figura 7 Diseño ventana principal

Cuando el usuario da clic en la entrada por voz, se despliega una ventana flotante en la cual se tiene un botón para iniciar con la detección de voz, para que después se identifique y se ejecute la petición del usuario. Esta interfaz se puede observar en la figura 8 donde algo a mencionar es que las ventanas que se vallan abriendo se ubicaran justo en la zona superior de la ventana principal para dar una mayor apariencia de ventanas flotantes como si se encontrase en un navegador web, pero en este caso es un software de escritorio. Esta ventana cuenta con un botón que activa el micrófono y etiquetas que muestran las respuestas que el asistente realiza.

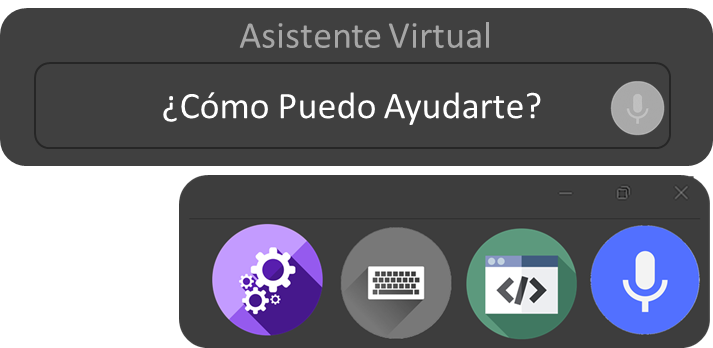


Figura 8 Diseño Interfaz de entrada por voz

La entrada por teclado como se analizó en el diseño lógico del asistente, tiene un funcionamiento similar al de la entrada de voz, teniendo la diferencia de que en este caso la petición es por teclado y, por ende, se requiere de un cuadro de texto para su inserción. Esta interfaz se puede ver en la figura 6, contando con componentes como etiquetas, cuadro de texto y un botón de envío.

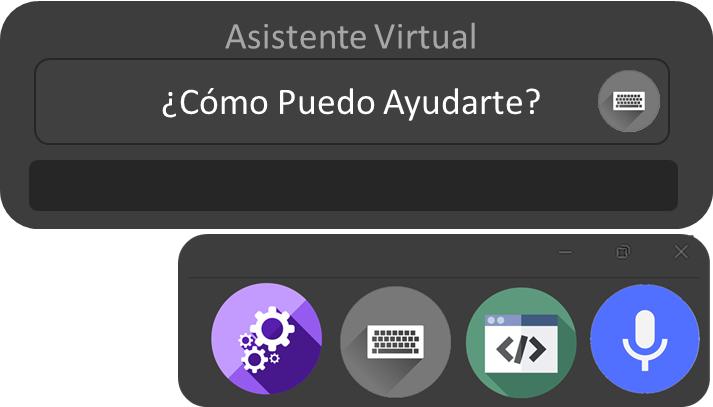


Figura 9 Diseño entrada por teclado

Cuando el usuario realiza una petición, el asistente identifica el mensaje recibido y ejecuta una búsqueda local o web para responder la petición del usuario y una vez determinada la respuesta, esta se despliega en una nueva ventana en la cual se encuentra el nombre de la consulta realizada y la información encontrada, para que el usuario pueda ver con claridad el resultado. Esta ventana que surgiría como respuesta de la petición se puede ver en la figura 11, la cual cuenta con elementos como etiquetas, cuadros de texto y un botón para adelantar a la siguiente consulta en caso de que el usuario quiera de otra respuesta diferente.

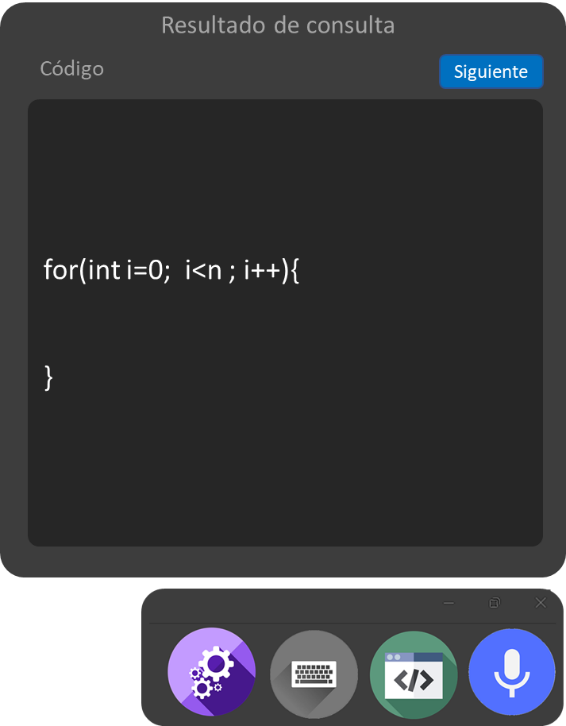


Figura 10 Diseño de ventana para el despliegue de resultados

El guardar el código de manera local o las consultas que realice el usuario puede ser muy útil para eficientar el tiempo y mantener información que sea de relevancia para el usuario, y es por ello que este software también cuenta con esta funcionalidad y una ventana dedicada para esta acción. En la figura 11 se muestra el diseño de la interfaz de inserción de código en el cual existe un cuadro de texto para el identificador de código y uno para el bloque de código que se va a almacenar, para que a través del botón de la ventana se ingresen los datos al fichero local si es que cumple con las características solicitadas.

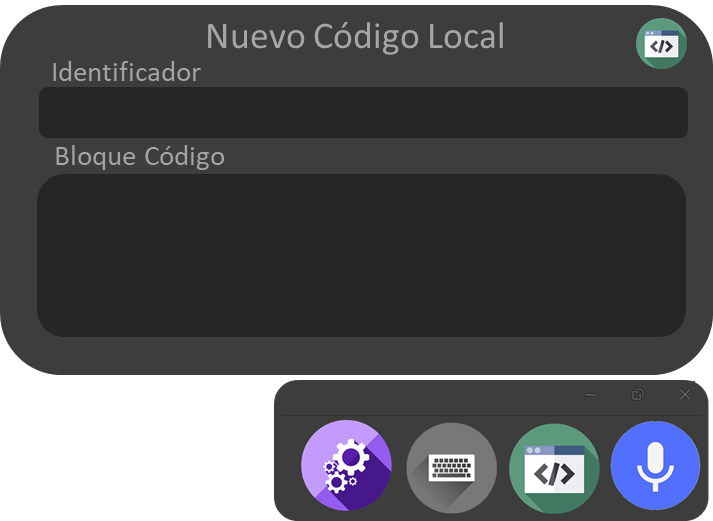


Figura 11 Diseño ventana inserción de código local

Este software desarrollado está planteado para que sea usado de manera constante, estando disponible para cualquier petición del usuario, por lo que, al ser una aplicación de uso constante es importante que se permita la configuración de atributos del funcionamiento de este, sea visuales o funcionales y es por ello que se implementa la ventana de ajustes con diseño como el de la figura 12, en la cual se pueden desarrollar acciones como configurar el dispositivo de entrada de audio, ajustes de la salida de voz y los colores de la interfaz.



Figura 12 Diseño interfaz de configuración de software

**Diseño Modelo Vista Controlador**

El modelo vista controlador, es un patrón de diseño que permite separar la base de datos (Modelo) la vista (GUI) y el controlador que permite la conexión entre los elementos previos. En este caso, el modelo son los archivos almacenados localmente, los cuales se consultan constantemente con funciones programadas y se despliegan a la vista para que el usuario acceda a la información de manera clara. En la figura 13 se representa la distribución del asistente a través de este patrón de diseño.

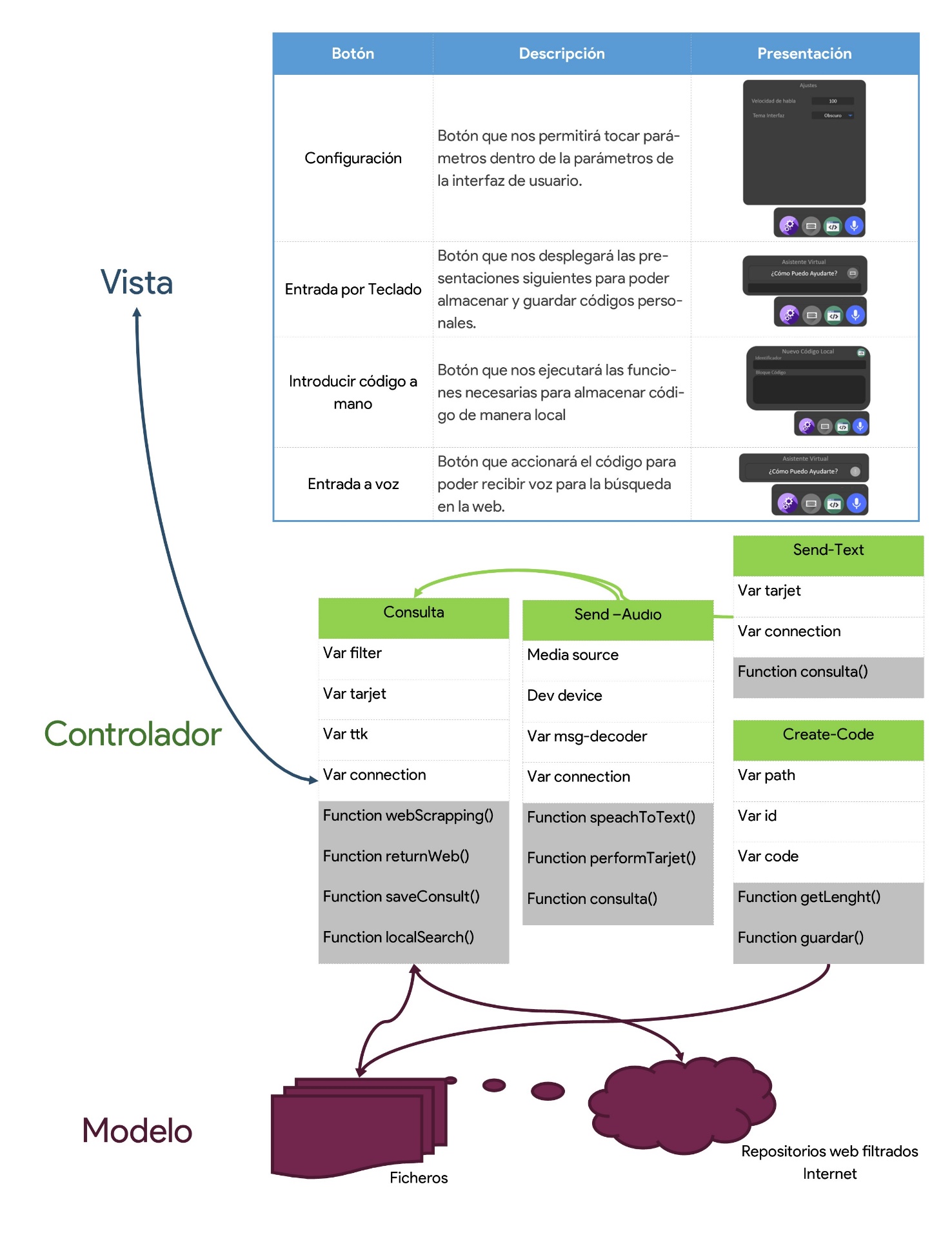


Figura 13 Modelo vista controlador (MVC)

De acuerdo con nuestra arquitectura MVC (Modelo – Vista – Controlador) la vista tendrá una interfaz minimalista que cuenta con cuatro funciones en una ventana reducida, ubicada (de manera inicial) en la esquina inferior derecha, las funciones de cada uno de los botones es la siguiente:

1. **Configuración:** La configuración hace referencia a los ajustes visuales que se le pueden aplicar a la interfaz, como lo es el tema y los colores que tiene, aspectos de la voz y el micrófono, como lo es, la reducción de ruido y la velocidad del narrador.
2. **Entrada por teclado:** El teclado tiene una función fundamental para dispositivos (sobremesa) que no tengan micrófono de manera nativa, ya que permite utilizar el asistente sin la necesidad de ingresar voz.
3. **Introducir código a mano:** El poder almacenar código de manera local es una función que se desea almacenar para poder ampliar los horizontes del usuario ya que, al introducir el código a mano, se abre la posibilidad de que el usuario comience a crear sus propias funciones que se adaptan a las necesidades de cada usuario.
4. **Entrada por voz:** Es la función principal a la que está orientada el asistente, a que permitirá al usuario usarlo mientras se ejecuta otro programa, sin necesidad de escribir con el asistente, agilizando el desarrollo por parte del usuario.

El controlador tendrá diferentes funciones que permitirán conectar con el modelo, procesar la información y posteriormente desplegarlos al usuario, al tiempo que el usuario también podrá almacenar en el modelo con ayuda de las funciones específicas en el controlador para guardar de manera local el código consultado o el código escrito por el usuario. Dentro de las funciones que destacan están:

1. **Consulta:** Que tiene como parámetros el *Filter* que hace referencia al conjunto de páginas que fueron seleccionadas para poder hacer la búsqueda de la información para ser mostrada en la aplicación, posteriormente se tiene el *Target* que es el objetivo de la búsqueda, en este caso el conjunto de palabras clave previamente analizadas y validadas, después se encuentra el *TTK* que es el tiempo para determinar si el usuario está conectado a internet y una vez validado podrá tomar dos caminos dependiendo el caso, webScrapping (online) o localSearch (offline), para finalizar, el elemento *connection* que se refiere a la conexión con nuestro modelo local (fichero) y en línea (web).

También contará con funciones que permitirán hacer la búsqueda mediante parámetros en internet *webScrapping* así como funciones para retornar el resultado (*returnWeb*) o guardar el código de manera local (*saveConsult*).

1. **Send-Audio, Send-Text:** En principio comparten la misma lógica, con la diferencia que el audio utilizará un par de módulos extra, los cuales son *speachToText* y *performTarjet*, que se encargarán de convertir la voz a texto y a encontrar las palabras clave de la consulta.
2. **Create-Code:** El principal uso de esta función es permitir que el usuario guarde su código de manera local sin importar si está conectado a internet o no, ya que se almacenará en el fichero local establecido. Recibirá parámetros como ruta del fichero (path), identificador de código (id) y el código como tal (code), a la vez que se desarrolla una función para obtener el largo del código para poder optimizar la búsqueda local (getLenght).

Para finalizar el modelo, que se compone de dos conexiones principalmente, la local (fichero con los códigos almacenados) y la remota (sitios que se encuentran en internet que cumplan con las características establecidas).

**Diseño de Acceso a los ficheros del software**

Como se explicó en la modelo vista controlador, este software hace uso de ficheros como modelo, por lo cual es importante definir como se realizará la comunicación para acceder a ellos y desde que interfaz el usuario estaría manipulándolos de manera transparente siendo que las funciones del programa lo harán de manera automática.

Los ficheros que se requieren para que el asistente cumpla con su funcionamiento se pueden ver en la figura 14, en los cuales se puede analizar que se requieren 5 tipos de fichero, el primero es el archivo de configuración del software, el cual se consultara al inicio de la ejecución del asistente para efectuar las configuraciones guardadas por el usuario y se editara en caso de que el usuario efectué algún ajuste en la ventana de configuración, también se tiene un archivo de Diccionarios, el cual se utiliza cuando se obtiene la petición del usuario, para realizar diversas comparaciones y determinar qué es lo que se está solicitando, para después realizar la consulta sea local o de tipo web y con base a sus respectivos archivos consultar el código o definición solicitada. En el archivo de datos locales se ingresa el código para consultarlo de manera local, este se puede ingresar desde la ventana de inserción destinada a ello o bien a través de la ventana de resultados se puede dar en guardar y automáticamente se generan los campos para ingresarlo al archivo local.



Figura 14 Ficheros del asistente

Este diseño de la figura 14 es representado de manera general, ya que muchas funciones más harán uso de estos archivos, y bien el archivo Diccionarios puede ser dividido en algunos otros para facilitar el acceso a lo que el usuario requiera. Se plantea que estos archivos puedan crearse de manera automática con valores predefinidos en caso de que se borren o se dañen para evitar que el software quede en estado no funcional y garantizar un funcionamiento continuo.

**Formato del fichero Local**

Los ficheros en este proyecto cumplen el rol de una base de datos, ya que en estos se almacenará información de utilidad para mantener un buen funcionamiento en cada uno de los módulos del software, ya que cada uno de ellos usa uno o más ficheros. Uno de los ficheros más relevantes a utilizar es el de códigos locales del usuario, ya que es el que se le puede dar mayor utilidad por los constantes cambios que se le pueden realizar en caso de que el usuario desee mantener información almacenada localmente.

Es importante definir un formato para poder acceder de manera rápida a los datos almacenados dentro del fichero, en la mayoría de ficheros a utilizar siguen un formato de lista, accediendo a través del nombre y obteniendo al valor que tenga enfrente, sin embargo, el fichero de códigos locales es diferente y sigue el formato que se puede ver en la figura 15, en el cual se explica gráficamente cómo funciona. Los códigos se almacenan en 4 partes, el identificador que es la parte superior, el número de líneas del bloque de código (Contando las secciones) el cual servirá para extraer solo el bloque de código o saltar a otro identificador en caso de que no sea lo que se busca, para que de esta manera sea más rápido realizar la consulta de los códigos, por lo que este tipo de formato permite que solo se haga lectura del identificador y del número de líneas, ahorrando mucho tiempo.

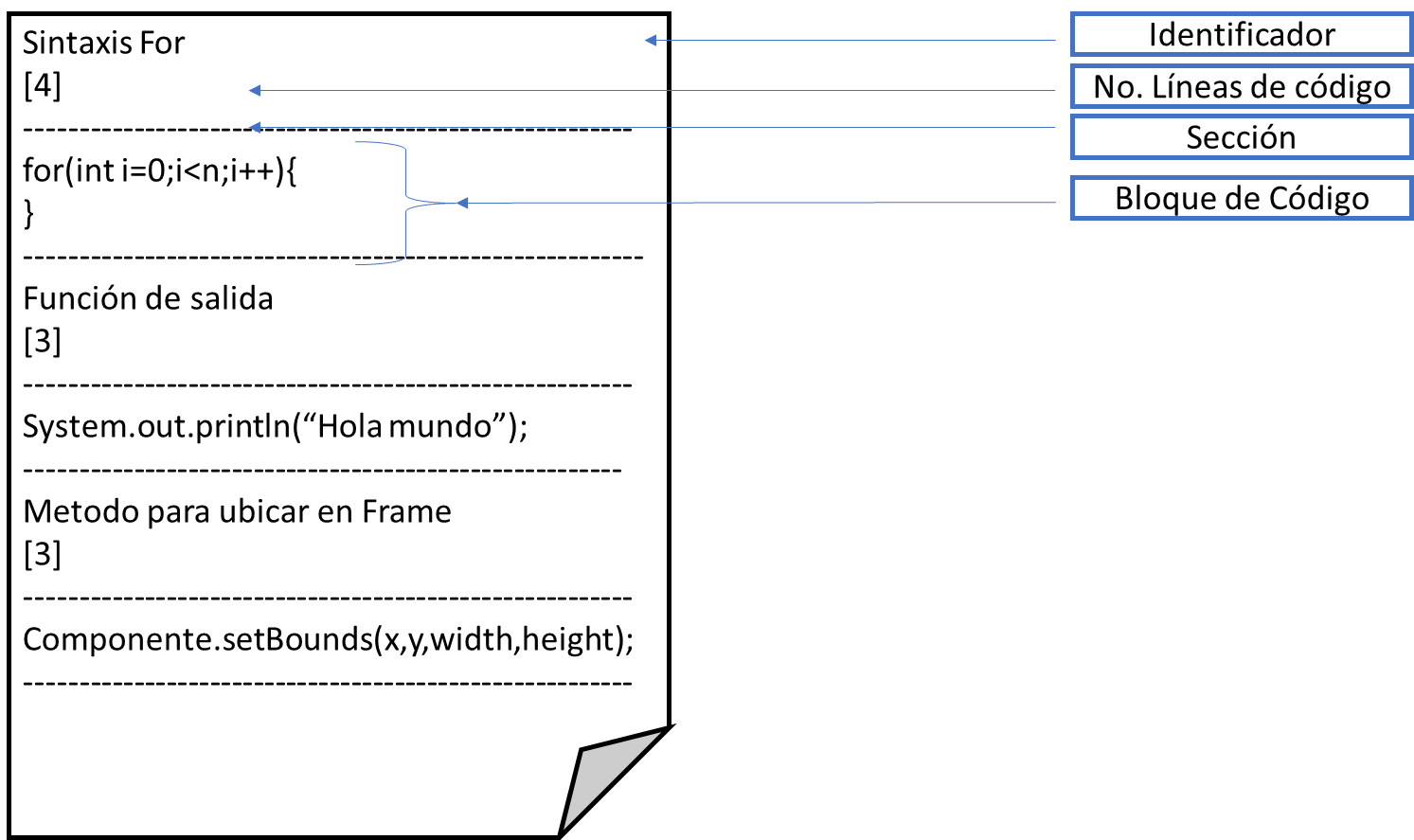


Figura 15 Formato del fichero local

**Diagrama de Clases**

En la figura 16 se puede observar cómo se estructura cada clase y cuál es la conexión entre ellas. El software estará divido en 6 clases principales que cumplen con una función en especifica y 10 que son subclases dentro de otra. La clase principal lleva como nombre *main*, y esta es la encargada de iniciar con la ejecución del programa, llamando a la clase Interfaz, la cual cuenta con alrededor de 10 subclases las cuales heredan cada una de la clase QWidget para la creación de las ventanas de la interfaz, cada una de estas clases construye los propios componentes que tienen dentro, como lo son los botones, cuadros de texto, etiquetas etc., conectando con sus funciones para cumplir con su función en caso de que el usuario tenga interacción con ellos.

La interfaz se crea primero, sin embargo después de que el usuario hace alguna acción se pueden mandar a llamar las demás clases, como lo es la de reconocimiento que tiene la función de realizar acciones como el reconocimiento de voz a través de la API de Google, la síntesis de voz con base a pyttxs3, o bien la identificación de la petición que el usuario ingreso con base a palabras que el asistente reconoce para realizar su acción y así decidir entre realizar una búsqueda Web, una búsqueda local o ayudar al usuario de manera local (Estos métodos de ayuda están dentro de la misma clase), sin embargo las consultas web y locales se apoyan de otras clases, las cuales son *WebScraping* y *IngresarLocal*, las cuales cuentan con métodos específicos para realizar las consultas. Cabe mencionar que la clase *config* tiene comunicación con todas las demás ya que es el que brinda cual es la configuración de la aplicación por el usuario.

La clase main únicamente se usara para llamar a la clase Interfaz y básicamente esta clase es la que se ejecutara en el hilo principal de la aplicación, la cual llamara a las demás clases que se ejecutaran bajo el mismo hilo de procesamiento, únicamente las clases Reconocimiento y WebScraping se ejecutaran en hilos externos, ya que estas clases procesaran información que requieren un poco mas de tiempo, ya que en el caso de reconocer la voz o realizar una búsqueda en la web, se toma un tiempo dependiendo de la velocidad de internet con la que se cuente en la maquina anfitrión. Cada una de las clases del programa se mandan a llamar entre si cada que requieran alguna opción o método que el objeto posea.

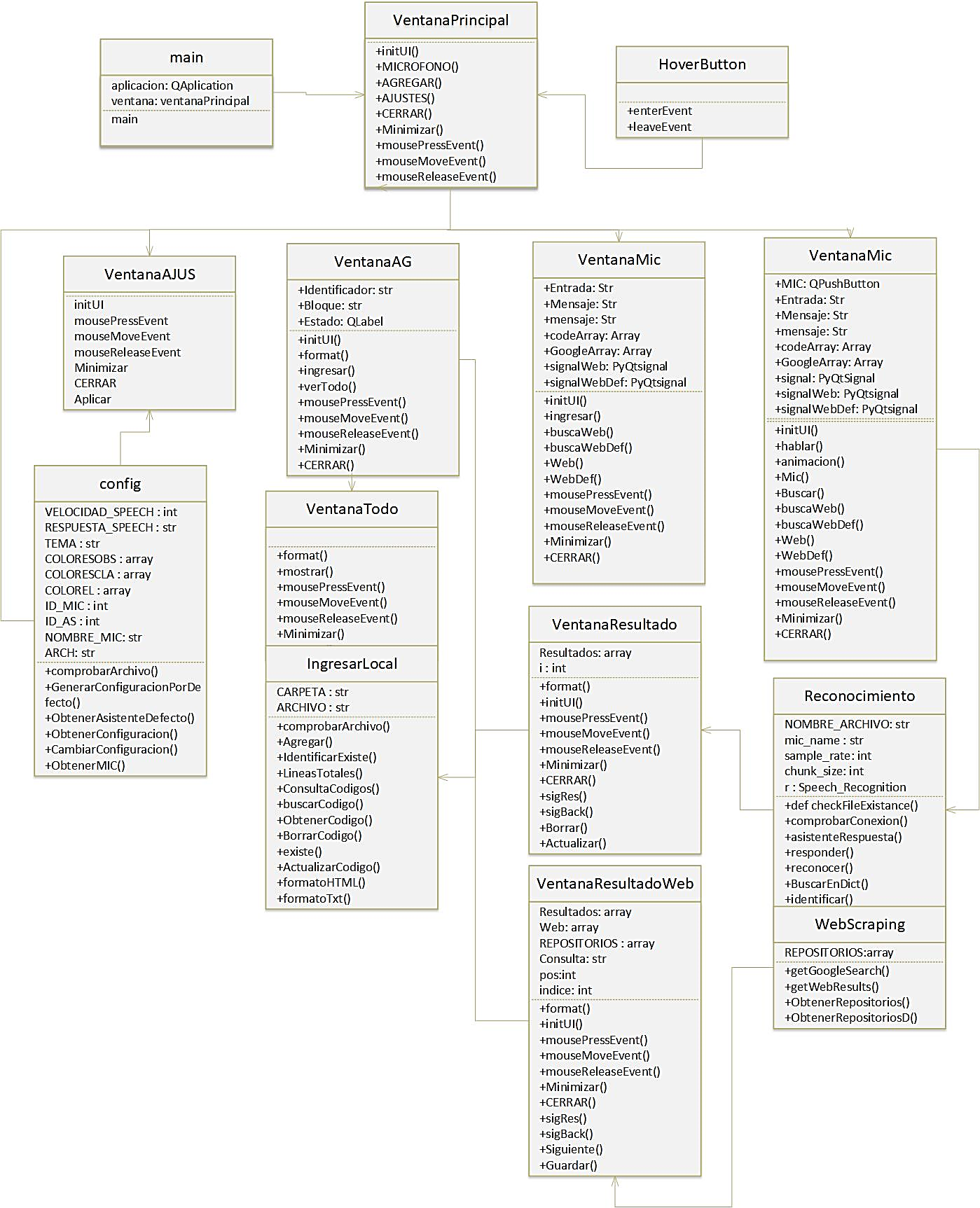


Figura 16 Diagrama de clases

### 3.4.2 Implementación

Una vez determinados los requerimientos y realizar el diseño visual y funcional del software es momento de realizar su implementación a través de las herramientas seleccionadas para su desarrollo. El lenguaje de programación de alto nivel Python, tiene características bastante relevantes que permiten el desarrollo de proyectos de gran escala y es muy utilizado en diversos ámbitos como lo es el desarrollo del backend en entornos web, es usado en ambientes dedicados a la inteligencia artificial y cuenta con gran cantidad de librerías que permiten desarrollar prácticamente cualquier cosa que el programador desee implementar, por lo tanto, este es el lenguaje elegido para la codificación de este proyecto.

Python por defecto cuenta con gran cantidad de librerías y funciones que permiten desarrollar proyectos correctos y funcionales, sin embargo, existen librerías que se necesitan descargar de manera externa que permiten del desarrollo de proyectos más avanzados, por lo cual es importante realizar su instalación previa. En la tabla 21 se enlistan todas las dependencias necesarias para la codificación del proyecto más una previa descripción, estas librerías tienen el propósito de desarrollar interfaces avanzadas, la implementación de la síntesis y reconocimiento de voz, así como la búsqueda por Web Scraping.

Tabla 21 Dependencias necesarias para Python

|  |  |
| --- | --- |
| **Nombre paquete** | **Descripción** |
| **PyQt5** | Es un set de módulos del framework Qt5 para la creación de interfaces graficas avanzadas |
| **PipWin** | Paquete que permite una instalación de paquetes con mayor compatibilidad para sistemas operativos Windows |
| **Speech Recognition** | Librería que permite el reconocimiento de voz a treves de diversas API’s como Google, Google Cloud, IBM, Sphinix etc. |
| **Pyttxs3** | Paquete para python para la conversión de texto a voz de manera offline. |
| **Pyaudio** | Proporciona enlaces para hacer uso de los dispositivos de entrada y salida de audio. |
| **PlaySound** | Permite la reproducción de archivos de audio a través de los distintos dispositivos de audio. |
| **Google** | Permite la conexión con servicios de Google |
| **GoogleSearch** | Permite el acceso y uso del buscador de Google para realizar consultas en la red. |
| **Beautiful Soup** | Extrae información con base a etiquetas específicas del HTML de un sitio consultado. |

Este software cuenta con un funcionamiento sencillo, que dependiendo de las entradas que, del usuario, el asistente realiza una acción especifica. En la figura 17 se representa el flujo que sigue para realizar cada acción, en la cual con base a la pulsación de un botón se puede realizar una acción especifica, como lo es realizar ajustes modificando ciertos parámetros del software, realizar una entrada por teclado para realizar una búsqueda, ingresar código de manera local al archivo almacenado en el disco, realizar una búsqueda por entrada de voz o bien cerrar la aplicación en caso de que no se use. Este flujo de instrucciones es el que se puede realizar a manera general, de manera específica, cada módulo sigue sus propios procedimientos para llegar al resultado esperado.

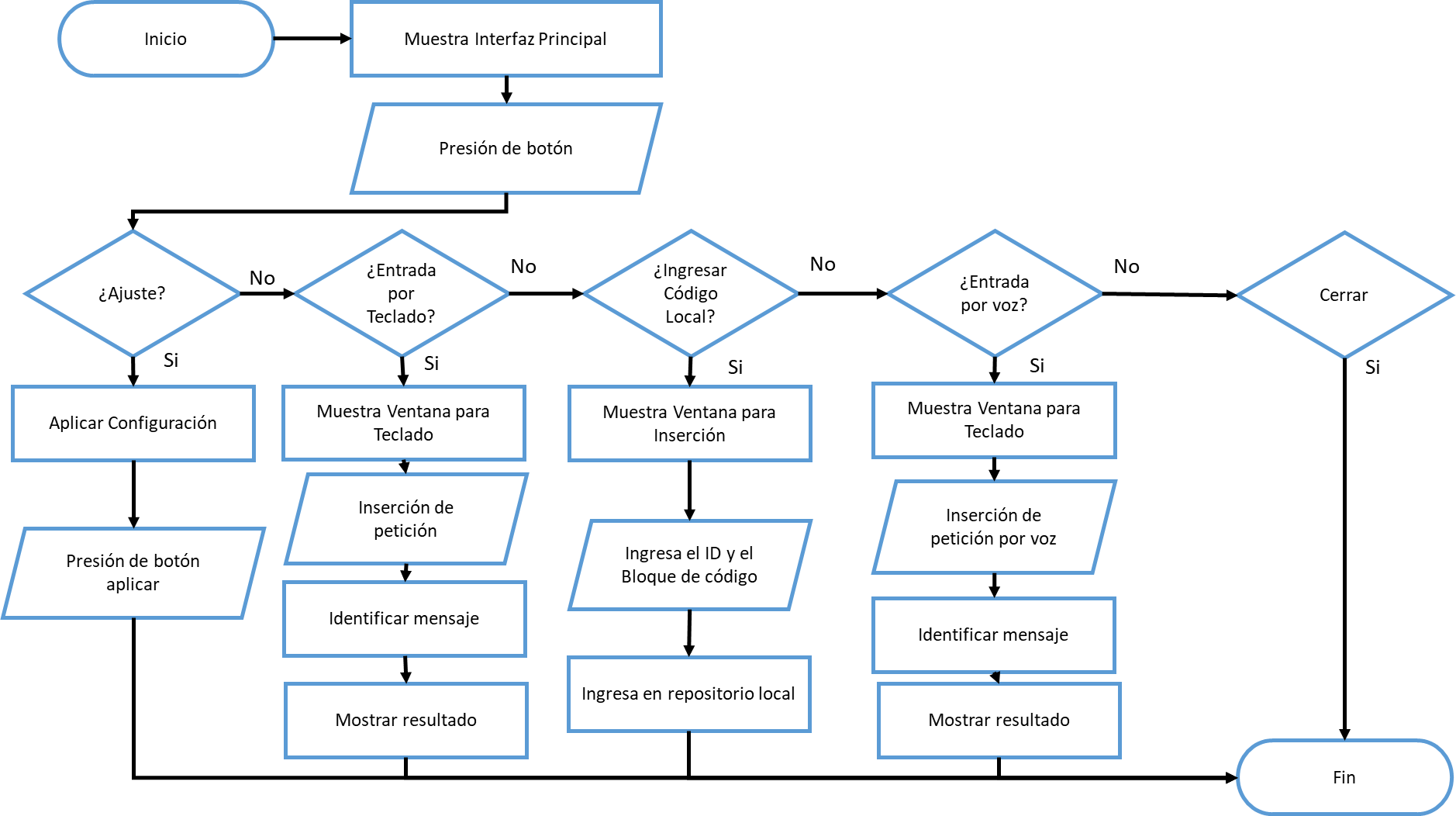


Figura 17 Diagrama de flujo del funcionamiento General

Una de las características de un asistente virtual es el uso del procesamiento del lenguaje natural, aceptando entradas a partir del teclado o por medio de la voz, en este caso se hizo énfasis en que se pudieran realizar las peticiones desde ambos medios. Para el reconocimiento del habla se hizo uso de la API de Google como anteriormente se ha mencionado, sin embargo, tanto la entrada por teclado y por voz siguen el mismo procedimiento para la identificación de palabras y la generación de una respuesta al usuario, lo único que cambia es el medio.

En la figura 18 se puede observar el flujo que sigue la identificación de las peticiones que realiza un usuario, donde primero se realiza la entrada por voz por medio del micrófono, para después con base a Google Speech realizar la conversión de voz a texto. Una vez obtenido el texto se convierte en mayúsculas para evitar problemas al comparar las diferentes cadenas que permiten la identificación. Una obtenido el texto se verificar que realmente se haya reconocido, después primero se realiza la comparación con un diccionario de palabras para realizar una búsqueda local, si no hay coincidencia se va a la comparación de palabras para búsqueda web, en caso de que no se verifica si es un saludo o una petición de ayuda, en caso de que alguno de estos si coincida devuelve una cadena para poder procesar la solicitud, limpiando las palabras que no sean de utilidad y realizar el siguiente proceso solo con los datos más importantes.

Cabe mencionar, que cada una de las palabras que identifica el asistente es con base a los diccionarios de frases que están almacenadas en archivos locales, los cuales se analizan línea con línea y se comparan con la entrada del usuario para determinar si existe alguna coincidencia, para que de esta manera el asistente pueda realizar la acción solicitada. El análisis de la entrada para determinar si coincide se hace con base a los caracteres que se tiene, se compara carácter con carácter entre la petición y el diccionario, haciendo una sumatoria si el carácter coincide, para que de esta manera con base al numero de letras totales de la cadena, se saca el porcentaje de las que fueron coincidentes, y si es que este porcentaje es mayor al 80%, se toma coincidencia y el asistente efectúe las acciones pertinentes con esa coincidencia. Esto se hizo con el motivo de que el asistente pueda predecir o entender peticiones que fueron escritas o de algún modo ingresadas erróneamente por parte del usuario.

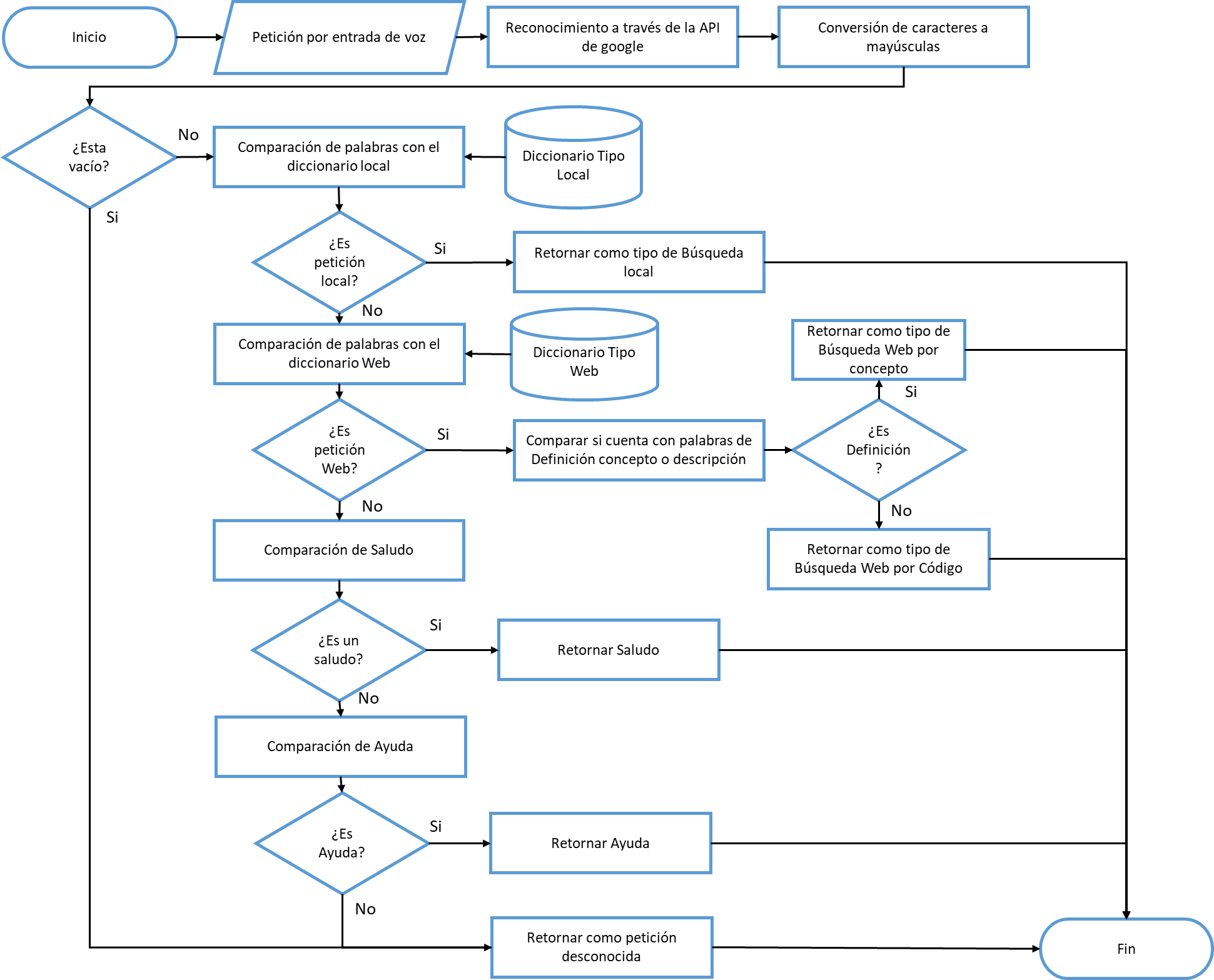


Figura 18 Diagrama de flujo del funcionamiento del reconocimiento de voz

La síntesis de voz del asistente tiene una implementación más sencilla que la del reconocimiento de voz, ya que únicamente requiere de una cadena de texto y se pasa como argumento en el método asignado para esa función, este proceso que se realiza se puede ver en la figura 19. La síntesis de voz depende de que el usuario la habilite o no, ya que con base a es parámetro una condición determina si se realiza o no el bloque de instrucciones que permiten la síntesis, el cual primero asigna las características del motor de texto a voz, se ingresa el texto a convertir a datos de tipo audio y se reproducen a través de un dispositivo de salida de audio. El reconocimiento de voz depende totalmente de que exista conexión a internet, por lo que si esta no existe, simplemente se notificara que no hay red, por lo que solo esta permitida la inserción de peticiones por teclado y solo se pueden realizar búsquedas locales.

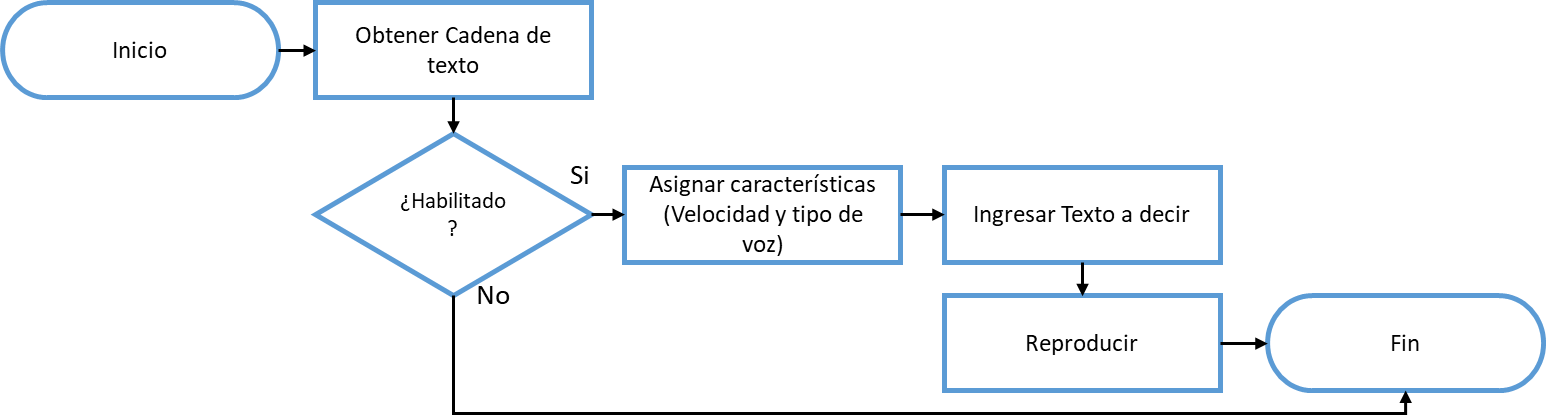


Figura 19 Flujo de instrucciones de la síntesis de voz

Para ingresar localmente un código para poder tenerlo a disposición en todo momento se requiere de un identificador y un bloque de código el cual se almacenará en una carpeta y un archivo almacenado en el mismo directorio del ejecutable. En la figura 20 se puede ver el proceso que se realiza para añadir códigos a estos archivos, en los cuales principalmente se verifica si la carpeta y archivo existen, en caso contrario los crea para después analizar el archivo, cabe mencionar que no se realiza un análisis línea por línea, ya que el fichero local hace uso de un formato que permite saltar líneas para ganar tiempo con base al número de líneas de los códigos ya almacenados y únicamente leer el identificador del código para determinar si el identificador ya existe y evitar valores duplicados, en caso de que no, se añade al final del archivo.

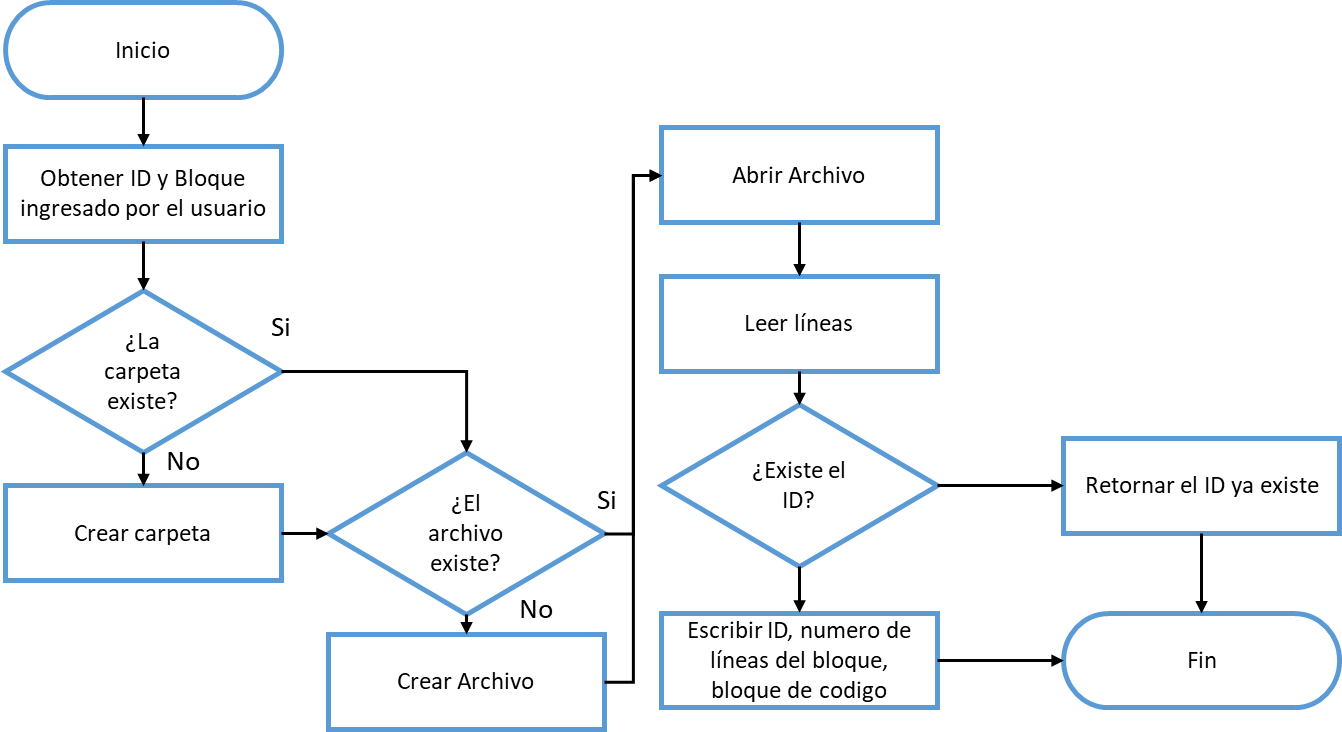


Figura 20 Manejo de inserción de código local

Al consultar el archivo local para obtener algún código que el usuario necesita, este se desplegara en una ventana donde el bloque de código se puede editar, con el fin de que con base a dos botones el usuario pueda actualizar el código almacenado o bien eliminarlo en caso de que no sea necesario, para realizar este proceso ambos tienen realización, ya que, para actualizar, primero se borra el código y se ingresa el nuevo.

Para borrar los códigos almacenados localmente se sigue el siguiente proceso:

1. Inicio
2. Abrir el archivo de códigos locales.
3. Leer el número de líneas del archivo y almacenarlas en una variable
4. Abrir nuevamente el archivo y leer realizando saltos de líneas con base al número de líneas por bloques de código.
5. Si el ID a borrar coincide con el archivo analizado, almacenar el ID, el número de líneas y el bloque de código en una variable de tipo cadena.
6. Remplazar el texto que coincida con la cadena anterior en la variable donde se almacenaron las líneas de todo el archivo.
7. Sobrescribir el archivo de código local con el resultado del remplazo del texto.
8. Fin

Para actualizar el código se hace uso del proceso de borrado y agregado, siguiendo la siguiente secuencia:

1. Inicio
2. Borrar el código con base al ID.
3. Agregar el nuevo código con el ID y su bloque de código al final del archivo.
4. Fin

Para la búsqueda de datos por internet se hizo uso de técnicas de web scraping a partir del buscador de Google y beautifulSoup para obtener la información de las etiquetas del HTML. Este algoritmo se puede ver en la figura 21, en el cual primero se identifica el mensaje, para después identificar que sea una búsqueda de tipo web y con base a los repositorios que están en listados en ficheros, se realiza la búsqueda en aquellos sitios, tomando las dos primeras coincidencias y realizando la inspección para retornarla al usuario.

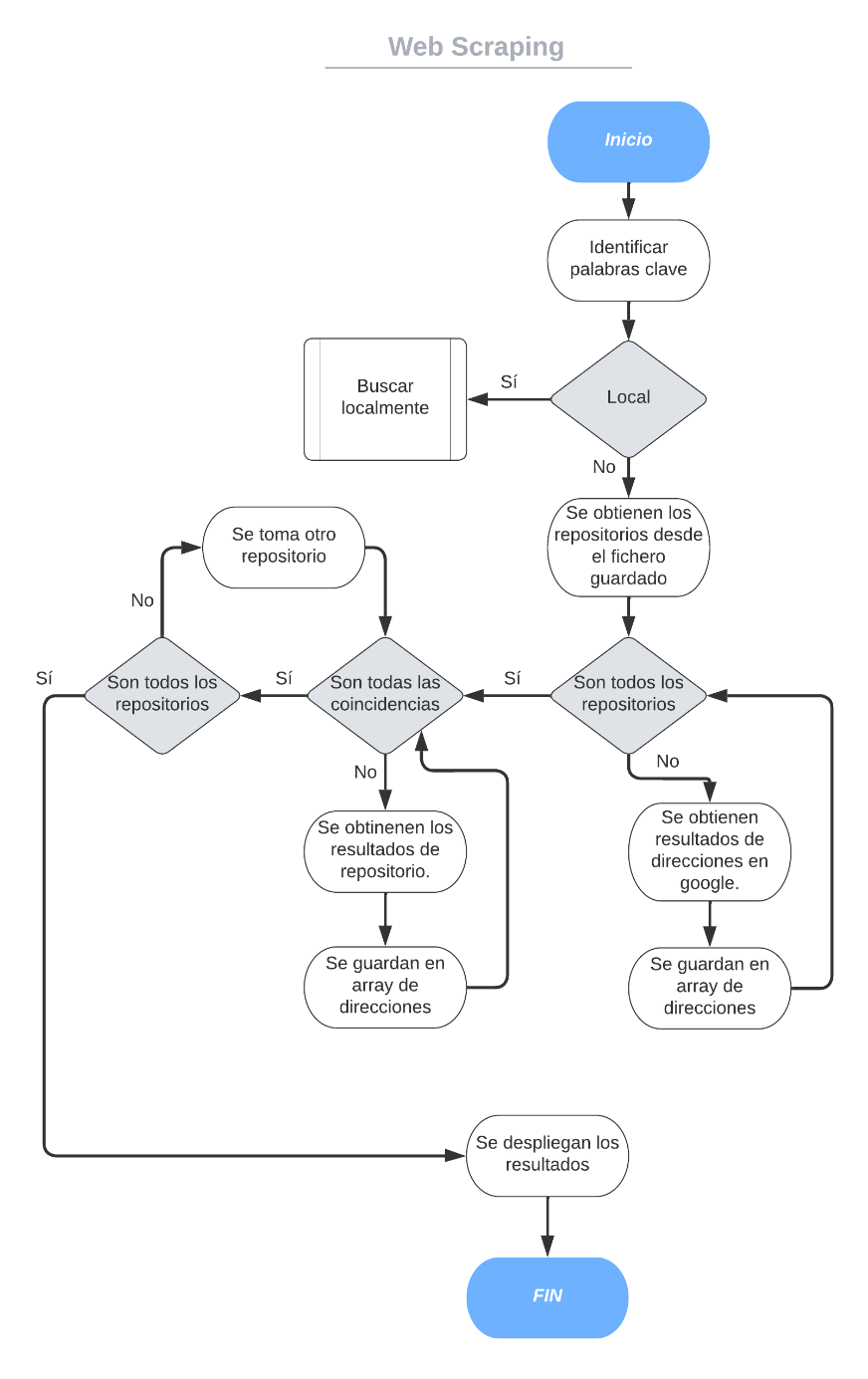


Figura 21 Algoritmo de Web Scraping

La codificación de la interfaz gráfica se desarrolló con base al framework Qt5 a través de los módulos de la librería PyQt5, la cual permite el desarrollo de componentes gráficos avanzados, haciendo uso de tecnologías como CSS y HTML. La interfaz implementada siguió los prototipos de diseño anteriormente creados, pero mejorados hasta cierto punto, siendo que cada una de las ventanas cuenta con una barra de acciones integrada y propia para realizar el cerrado y minimizado, esto con el propósito de que la interfaz no sea afectada visualmente cuando se tiene otro tema o una versión diferente del sistema operativo, evitando errores de visualización.

Cada una de las ventanas están creadas a partir de la clase QMainWindow y QWidget de Qt5, con propiedades independientes del gestor de ventanas del sistema operativo, lo cual tiene como desventaja que no se pueden hacer ciertas acciones con la ventana, pero estas se implementaron a través de la codificación de manera básica para que se pueda ocupar sin problemas. En la figura 22 se puede ver el resultado obtenido en cuanto al aspecto gráfico, teniendo como característica el uso de bordes redondeados y el uso de colores amigables para la vista.

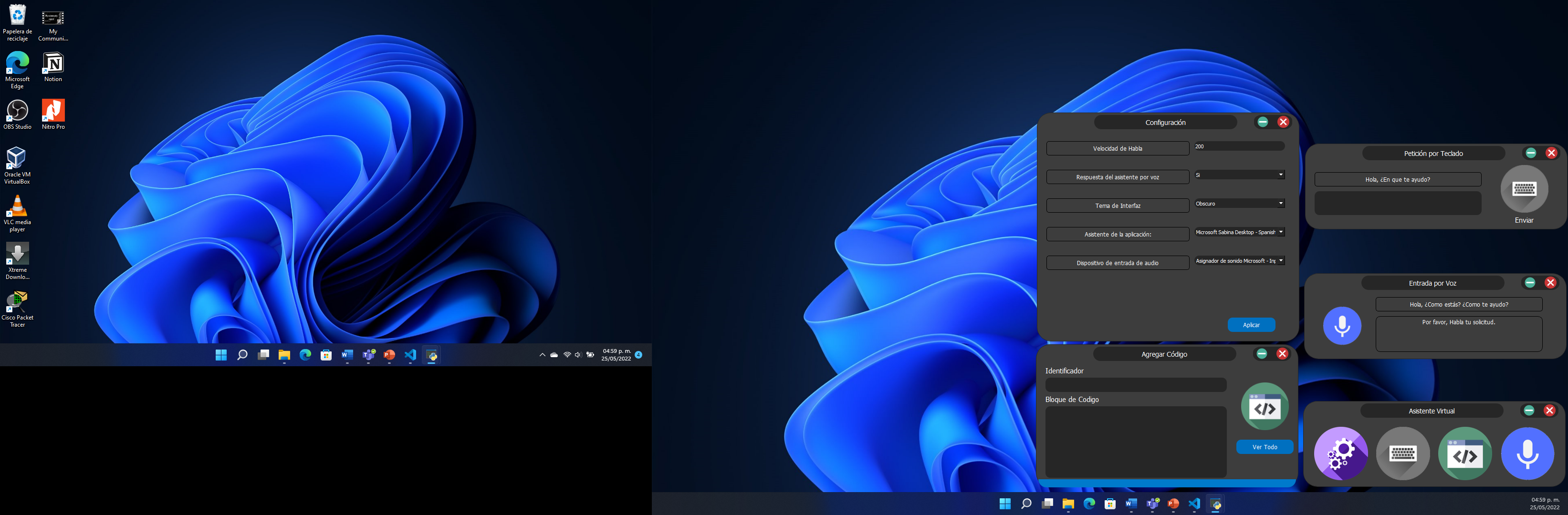


Figura 22 Codificación de la interfaz Gráfica

Al ser una aplicación orientada a programadores y a la consulta de información como lo son conceptos, sintaxis o algoritmos, se muestran códigos los cuales no son visualmente amigables si se maneja un texto normal y de un solo color, por lo cual es importante el enfatizar en palabras claves o palabras reservadas dentro del código, para que sea más fácil y comprensible su lectura, por ello con base a los lenguajes de marcación HTML y CSS se programaron estilos que con base a cuadros de texto que están a la escucha de las acciones del teclado, se tome lo ingresado y seguido se de un formato a palabras reservadas, signos o símbolos asignándoles un color especifico y la figura 23 es ejemplo de esta implementación. Esto se presenta en todas las zonas donde se espera que se despliegue código, por ejemplo, al ingresar uno de manera local o consultar a través de la web o en el archivo local.

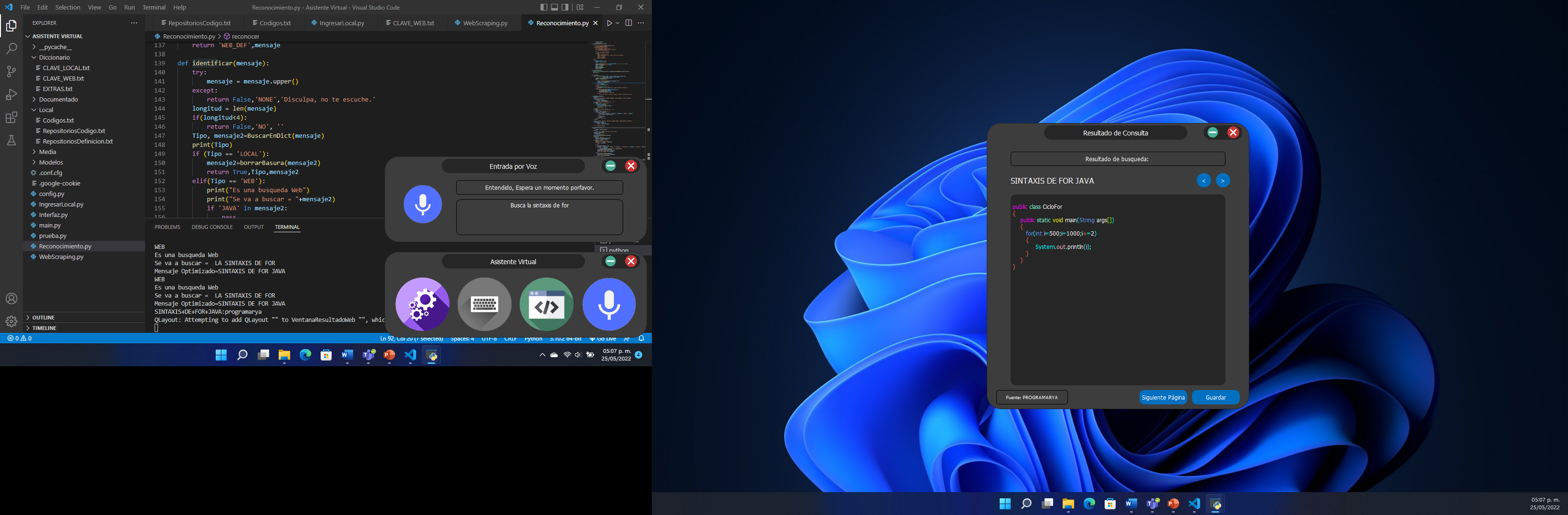


Figura 23 Formato de códigos en la interfaz

La codificación de un software, normalmente realiza instrucciones secuenciales, lo cual es adecuado cuando el tiempo de procesamiento es corto ya que las acciones tienen respuesta de manera inmediata y el flujo de trabajo es continuo, sin embargo, este tipo de programas da problemas cuando se requiere de procesar tareas que tomen demasiado tiempo, ya que mientras el programa está procesando la tarea, la interfaz se puede quedar en un estado de espera, sin que se pueda navegar a otras funciones ya que el programa está ocupando procesando la otra solicitud y esto a veces hace que el sistema detecte que el programa no responde debido a que la interfaz queda bloqueada.

Para solucionar esto, se requiere que las operaciones del programa no sean procesadas de manera secuencial, si no que sea divido, pudiendo ser a través de hilos o multiprocesos, los cuales se ejecutaran de manera independiente permitiendo que la interfaz se mantenga en el hilo principal con acceso a todas las funciones y el procesamiento de consultas, entradas de voz o la síntesis de voz se realicen en hilos o multiprocesos de manera independiente, agilizando mucho la experiencia de usuario. El uso de hilos o multiprocesos depende mucho de lo que el programador está buscando, ya que ambos tienen sus ventajas y desventajas, pero en este caso se hizo uso de ambos, hilos para el reconocimiento de voz y la búsqueda web, y multiprocesos para la síntesis de voz, ya que al reproducir el sonido a veces puede afectar los hilos padres, por lo que un multiproceso que se implementa de manera independiente es perfecto ya que no interfiere.

En la figura 24 se puede ver un ejemplo de cómo se implementan los hilos de procesamiento en el software, siendo que el hilo principal que es la interfaz de usuario crea el hilo con base a la petición del usuario, sea reconocimiento de voz, búsqueda en la web o bien para la síntesis de voz el cual hace uso de multiprocesos, pero sigue el mismo patrón de funcionamiento. Al hilo hijo creado se le asignan parámetros a utilizar para sus procesos y el resultado que obtenga lo regresa al hilo padre para que pueda desplegar el resultado en la interfaz o bien ejecutar alguna serie de instrucciones.

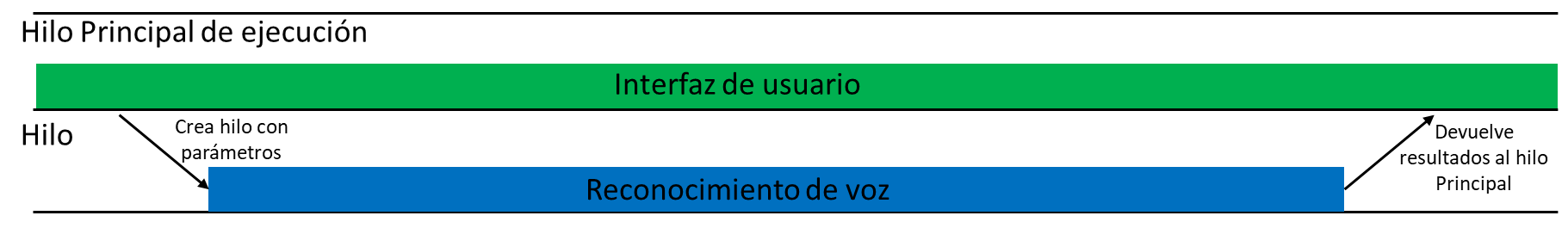


Figura 24 Implementación de hilos en el programa

Esto permite que el programa se pueda utilizar de manera constante sin interrupciones que afecten el funcionamiento del software y evita que el sistema operativo detecte falsas alarmas de algún error en el funcionamiento.

Al hacer uso de hilos, se pueden realizar consultas, aunque ya se esté procesando alguna otra solicitud, mostrándose al final el que se haya procesado primero. Una vez desarrollado cada uno de los módulos necesarios y codificado cada una de las interfaces, el resultado final es el que se muestra en la figura 25, pudiendo realizar el procesamiento del lenguaje natural vía teclado o por medio de la voz para después realizar una consulta a través de la web con base a técnicas como web scraping o bien una consulta en el archivo local.

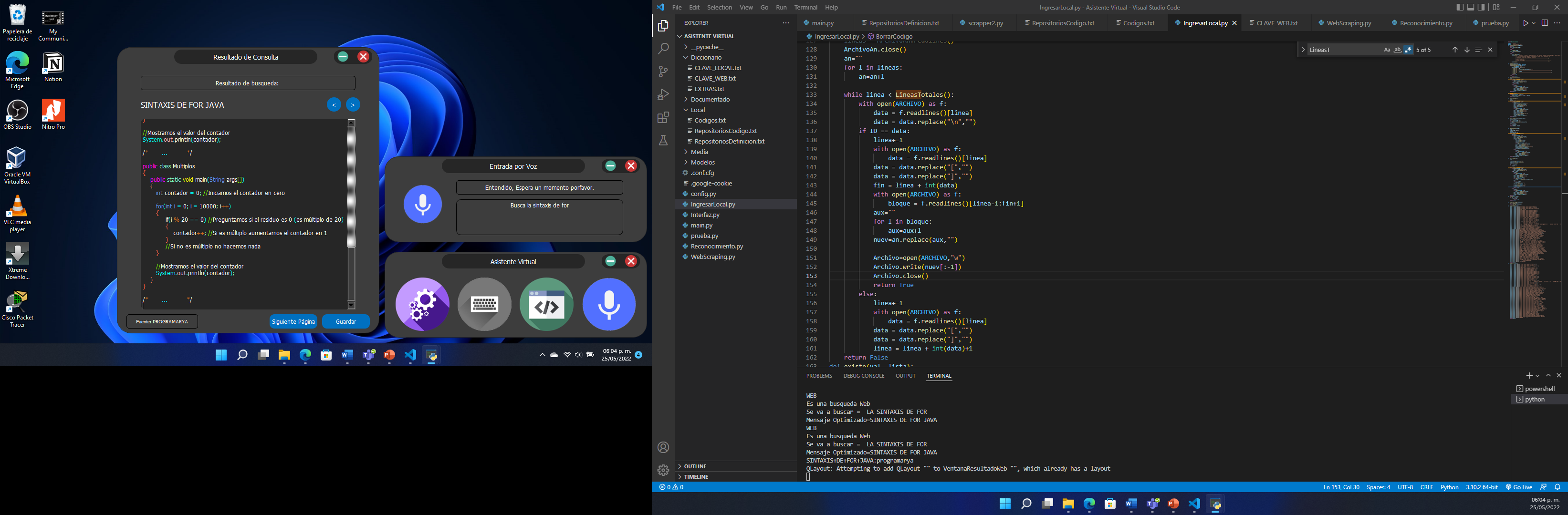


Figura 25 Implementación del software

### Pruebas

En la tabla 22 se describen los resultados obtenidos en algunas de las pruebas generales realizadas al asistente virtual con relación a la búsqueda de sintaxis, en posteriores tablas se puede encontrar también los resultados de las búsquedas de definiciones, ayuda al usuario, búsqueda local y un concentrado de búsquedas aleatorias.

Tabla 22 Pruebas de búsqueda web con respecto a sintaxis de funciones

|  |  |
| --- | --- |
| Búsqueda | Resultado |
| Ciclo for | Satisfactorio |
| Ciclo while | Satisfactorio |
| Sentencia if | No satisfactorio |
| Sintaxis para imprimir | Satisfactorio |
| Sintaxis para convertir decimal a binario | Satisfactorio |
| Sintaxis para convertir de decimal a hexadecimal | Satisfactorio |
| Sintaxis para convertir de decimal a octal | Satisfactorio |
| Sintaxis para crear un hilo | Satisfactorio |
| Sintaxis para crear un arreglo | Satisfactorio |
| Sintaxis para crear un socket | Satisfactorio |

La tabla 23 muestra los resultados obtenidos con la búsqueda de definiciones.

Tabla 23 Pruebas de búsqueda Web con respecto a definiciones

|  |  |
| --- | --- |
| Búsqueda | Resultado |
| Definición de Programación | Satisfactorio |
| Definición de for | Satisfactorio |
| Definición de ciclos | Satisfactorio |
| Definición de condicionales | Satisfactorio |
| Definición de sockets | Satisfactorio |
| Definición de base de datos | Satisfactorio |
| Definición de hilo | Satisfactorio |
| Definición de arreglo | Satisfactorio |
| Definición de matriz | No satisfactorio |
| Definición de tipo de dato | Satisfactorio |

La ayuda en un total de 20 veces solicitada solamente se acertó en un 75% de las ocasiones (15 de 20).

De un total de 40 peticiones random se contestaron satisfactoriamente 38, obteniendo un 95% de eficiencia.

Los resultados en la imagen 26 reflejan que el consumo en estado de reposo de nuestro asistente virtual es de 0%, así como el uso de la memoria RAM es de apenas 29.1 MB, no tiene lectura ni escritura en disco y tampoco conexión con la red.

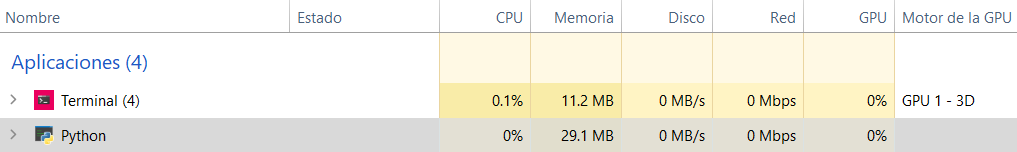


Figura 26 Consumo en repeso del software

Una vez que se realizan las peticiones de alguna consulta de información se obtiene que en la memoria RAM aumentan 10 MB obteniendo así 40 MB, la escritura ahora es de 100 KB y el internet no consume nada registrable.

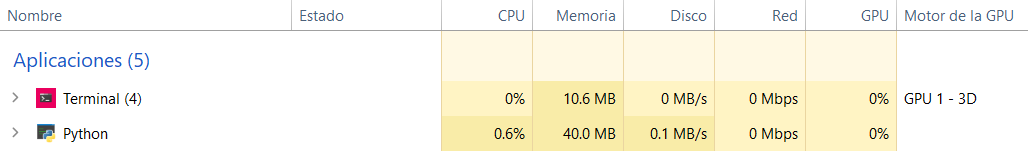


Figura 27 Consumo al ejecutar petición

Cuando está activo el procesamiento de la voz la escritura en disco se triplica obteniendo 300KB y se aumentan 3.4MB en la memoria RAM, se obtienen picos en el procesador de hasta 6.3%, hablando de un procesador de 8 núcleos y 16 hilos con una frecuencia base de 2.9GHz, se estaría ocupando un núcleo con dos hilos y cerca de 0.1820GHz

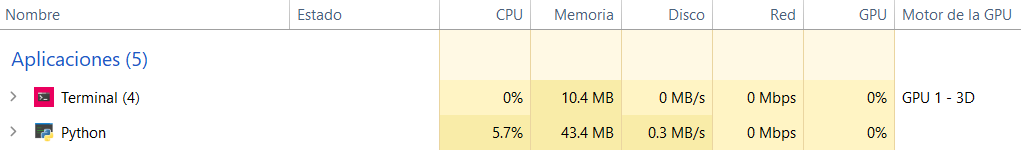


Figura 28 Consumo al realizar procesamiento de voz

# IV. Resultados obtenidos y discusión

El software desarrollado cumplió correctamente con las funciones de consulta planteadas, sin embargo, para comprobar si se logró verificar si la hipótesis se realizaron diversas pruebas con diferentes tipos de ambientes.

La primera prueba se realizó con un muestreo de 30 situaciones en las que se realizaron consultas por internet sobre conceptos y sintaxis referentes a la programación de java, y como se ve en la figura 11, en el 90% de estas se tuvieron resultados satisfactorios, con una tasa de error del 10%.

Figura 29 Pruebas de la precisión de búsqueda web

La consulta por internet tiene la precisión deseada, por lo cual se prosiguió con realizar pruebas de la detección de la voz con diversos parámetros, en la figura 30 se puede apreciar que en condiciones óptimas con poco ruido en el ambiente la precisión es del 93% en el reconocimiento.

Figura 30 Pruebas de reconocimiento de voz con condiciones optimas

Cuando las condiciones de ruido son malas en el ambiente, los resultados se hacen muy poco precisos siendo que el error llega a aumentar dando como resultado una tasa de error de hasta el 60%, teniendo tan solo una precisión de solo 40% como se puede ver en la gráfica de la figura 31, por lo que es importante que el uso del software este en condiciones aceptables o bien, es necesario implementar métodos para reducir el ruido captado por el micrófono como la llama cancelación de ruido.

Figura 31 Pruebas de reconocimiento de voz en condiciones de ruido en el ambiente

Un factor igual de relevante a la hora de implementar el procesamiento del lenguaje natural, es el reconocer el habla sin importar con qué tipo de emoción lo mencione o a qué velocidad realice la pronunciación de las palabras, por ende, se realizaron de igual forma pruebas con cambios en la velocidad a la hora de que el usuario dicte una petición y como se ve en la figura 14, la precisión que logra a la hora de reconocer una petición a gran velocidad es del 70%, reduciendo hasta un 20% que cuando se presenta la consulta en condiciones normales.

Figura 32 Pruebas de reconocimiento de voz con petición dictada a gran velocidad

# V. Conclusiones

La tecnología debe ser utilizada para el crecimiento del conocimiento de la humanidad, en este caso el asistente inteligente para el aprendizaje del lenguaje java cumplió con los objetivos, como la búsqueda en la web mediante reconocimiento del lenguaje natural y su procesamiento, de manera local con los ficheros implementados y sus respectivos identificadores, así como resultados más que aceptables en el conjunto general de funcionalidades implementadas.

El presente proyecto tuvo el objetivo principal de desarrollar un asistente que apoye a los estudiantes que desean aprender el lenguaje de programación java, por lo cual, este software estaba más enfocado en realizar la búsqueda de información por internet o localmente con una precisión del 90% en los resultados para que el usuario obtenga la información que le sea de ayuda, y con base a las pruebas realizadas este porcentaje de precisión se alcanzó adecuadamente, con un reconocimiento del habla del 93% en condiciones adecuadas.

Los puntos a mejorar de este proyecto son especialmente en la validación del reconocimiento en condiciones donde no sean favorables para el asistente ya que por ejemplo en la captación de voz con ruido de fondo la tasa de error aumento hasta en un 60% lo cual no es nada optimo. Otro aspecto a considerar es la implementación de algún motor de reconocimiento de voz offline en un futuro para que el asistente pueda entender la voz sin importar que no tenga conexión a internet el dispositivo, esto a través de tecnologías como por ejemplo Sphinix, siendo que sería de gran utilidad ya que el asistente puede realizar consultas de manera local con base al fichero que almacena esos datos.

Se puede concluir que el asistente virtual basado en procesamiento de lenguaje natura para el aprendizaje del lenguaje Java se realizó correctamente, sin embargo, hay detalles que pueden ser mejorables y que en futuras versiones se pueden corregir o bien escalar el proyecto a más lenguajes de programación ya que las bases de java se pudieron implementar de una manera correcta y funcional.

# Fuentes de información

1. J. L. Oropeza, “Algoritmos y métodos para el reconocimiento de voz en español mediante sílabas”, Articulo científico, Computación y sistemas Vol. 9, CIC-IPN, 2006, pp. 270-286.
2. P. Mauricio, “Asistente Inteligente basado en Agentes para modelar Sistemas mediante el uso de UML”, trabajo de titulación, Universidad Nacional Experimental de Guayana Ciudad Guayana, Estado Bolívar, Venezuela, 2008.
3. B. Manuel, F. Baltasar, V. Antonio, “Un asistente inteligente para UNIX basado en la documentación”, Revista de la asociación para el desarrollo de la informática educativa, pp. 25-35, 2010.
4. D. Mauricio, G. Andrea, M. Néstor, M. Patricio “Asistentes virtuales en la educación universitaria”, Proyectos de investigación de la undécima jornada de tecnología aplicada a la educación matemática universitaria, Buenos Aires, Argentina, 2011.
5. G. A. Martínez, G. A. Torres “Reconocimiento de voz basado en MFCC, SBC y Espectrogramas”, Articulo científico, Revista de ciencia y tecnología INGENIUS, 2013, pp.12-20.
6. C. Fernández, “Desarrollo e implementación de un asistente virtual para Linux”, Trabajo de fin de grado, Universidad politécnica de Valencia, España, 2014.
7. M. Alién, “Asistente de aprendizaje personal. Un nuevo paradigma en la educación”, universidad de matanzas, 2016.
8. A. Rafael “Uso de asistente virtual para el aprendizaje de temas selectos de la física”, trabajo fin de master, Tecnológico de Monterrey, 2016.
9. A. Maza, S. Daniel, “Análisis, diseño e Implementación de un asistente tutor basado en computación cognitiva para su uso en entornos visuales de aprendizaje”, trabajo de titulación, Área técnica, Universidad técnica particular de la roja, 2017.
10. G. Juan, "Algoritmos y programación para docentes”, Fundación Gabriel Piedrahita Uribe, Segunda Edición, Colombia, 2017.
11. V. Leyva, E. Jara, E. Riofrio, P. Teruel, “Facebook como herramienta para el aprendizaje colaborativo de la inteligencia artificial.” Didasc@lia: Didáctica Y educación ISSN, 2018, pág. 27-36.
12. G. Luis, “Asistente virtual tipo ChatBot.” Trabajo de titulación, Universidad Católica de Colombia, Facultad de Ingeniería, Bogotá, 2018.
13. D. Carlos, “Diseño e implementación de un asistente personal inteligente en español basado en reconocimiento de voz empleando Raspberry Pi”, Trabajo fin de grado, Universidad Carlos III de Madrid, 2018.
14. I. Erick, “Implementación de un asistente basado en inteligencia artificial para ambientes de aprendizaje de niños con discapacidad visual, Tesis doctoral, Facultad de filosofía y letras, Benemérita universidad autónoma de Puebla, 2020.
15. M. M. Echeverri & R. Manjarrés, “Asistente virtual académico utilizando tecnologías cognitivas de procesamiento de lenguaje natural”, articulo científico, revista politécnica, Colombia, 2020, pp. 85-96.
16. L. José y G. Ángel, “Programación orientada a objetos C++ y Java”, Instituto Politécnico nacional, Primera Edición, editorial PATRIA, México, 2014.
17. P. Gonzalo, “La inteligencia artificial y su influencia en la eficiencia del comercio internacional”, Tesis para titulación, Facultad de ciencias empresariales, Perú, 2020.
18. D. Glez, A. Lourencio & H. Lopéz, “Web scraping technologies in an API world”, Articulo cientifico, Briefing in bioinformatics Vol. 15. pp 788-79, Abril 2013.
19. M. Damián, “Redes Neuronales: Conceptos básicos y aplicaciones”, Reporte de investigación, Facultad regional rosario departamento de ingeniería química, pp. 8-11, 2001.
20. M. Dorys, C. Iván, G. Karolina, Q. Andrea, M. Christian, G. Teresa “Análisis del Estado Actual de Procesamiento de Lenguaje Natural”, Revista Ibérica de Sistemas e Tecnologías de Información, RSTI, N° E42, pp. 126-136, 2020.
21. S. Adrián, “Puesta a punto de un robot real autónomo inteligente usando ROS”, Trabajo de fin de grado, Universidad politécnica de Cartagena, ciudad de Cartagena Colombia, 2021.
22. C. Edwin, “Desarrollo de una aplicación de seguridad vial usando el Sistema de Posicionamiento Global (GPS); para prevenir accidentes vehiculares”, Tesis profesional, Universidad peruana Unión, ciudad de Juliaca, Perú, 2020.
23. Electric and Software. "Las 10 mejores herramientas de reconocimiento de voz para Linux". Electronics & Software. Disponible en: https://electronicssoftware.net/10-mejores-herramientas-de-reconocimineto-de-voz-para-linux/ (accedido el 28 de marzo de 2022)

# Anexos

Tecnológico de Estudios Superiores de Jocotitlán

Ingeniería en Sistemas Computacionales

Asistente Virtual Basado en Procesamiento de Lenguaje Natural para el aprendizaje del Lenguaje de Programación de Java.

Manual de Usuario

Julio Ponce Camacho y Angel Maldonado García

Desarrolladores

junio de 2022

Contenido

[Requisitos del sistema: 79](#_Toc106700897)

[Principales Funcionalidades 80](#_Toc106700898)

[1. Micrófono 80](#_Toc106700899)

[2. Agregar código local 80](#_Toc106700900)

[3. Entrada de texto 81](#_Toc106700901)

[a. Resultados de Búsqueda 81](#_Toc106700902)

[4. Ajustes 82](#_Toc106700903)

Tabla de Ilustraciones

[Ilustración 1 módulo de entrada por voz. 80](#_Toc106700904)

[Ilustración 2 Modulo: Agregar Código Local 81](#_Toc106700905)

[Ilustración 3 Modulo: Búsqueda por teclado. 81](#_Toc106700906)

[Ilustración 4 Resultados de la consulta. 82](#_Toc106700907)

[Ilustración 5 Modulo de configuración 83](#_Toc106700908)

# Requisitos del sistema:

1. Software:
   1. Python v3.10 o superior.
   2. Speach recognition v3.8.1 o superior (online).
   3. Pyttsx3 v2.90 o superior (offline).
   4. PyQt5 v5.15.6 o superior.
   5. Windows 10 o superior (versiones home, pro o single)
2. Hardware:
   1. Mínimo
      1. Procesador con 4 núcleos @ 3.5Ghz.
      2. 8 Gb RAM.
      3. 5 Gb de almacenamiento.
      4. Conexión a internet de 5Mbps.
      5. Micrófono pasivo incorporado al pc.
   2. Recomendado
      1. Procesador con 8 núcleos @ 4Ghz.
      2. 16 Gb RAM.
      3. 10 Gb de almacenamiento.
      4. Conexión a internet de 10Mbps.
      5. Micrófono pasivo incorporado al pc.

# Principales Funcionalidades

El asistente cuenta con 4 funciones, en las cuales se encuentran búsqueda por vox, búsqueda por texto, agregar código local y ajustes, cada uno de ellos presenta logos que hacen referencia a las funcionalidades antes mencionadas.

1. Micrófono**:** El micrófono es módulo que nos permite la búsqueda por voz, al seleccionar la opción, nos aparecerá un menú como el de la ilustración 1:

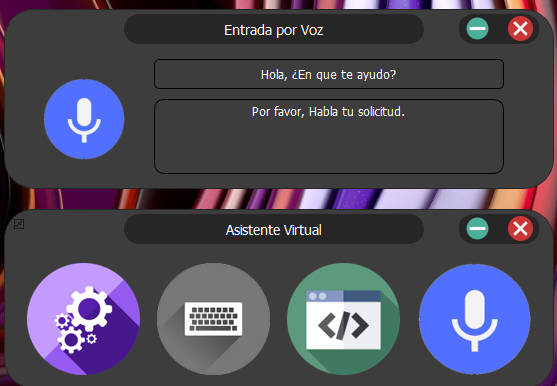
****

Ilustración módulo de entrada por voz.

Como podemos observar en la figura uno encontraremos un icono de un micrófono de color azul que al dar clic sobre él se pondrá de color verde, esta será la señal de que el micrófono está activo y en escucha de las peticiones que le hagamos.

1. Agregar código local**:** Para agregar código de manera local nos encontraremos con el menú de la ilustración 2, en donde encontraremos opciones como abrir los códigos que tengamos guardados. También encontraremos dos campos con los nombres de Identificador y con el nombre de Bloque de código, estos campos sirven para que el asistente sepa cómo llamaste a ese bloque de código (Identificador) y el código como tal que se guarda con ese nombre (Bloque de código).

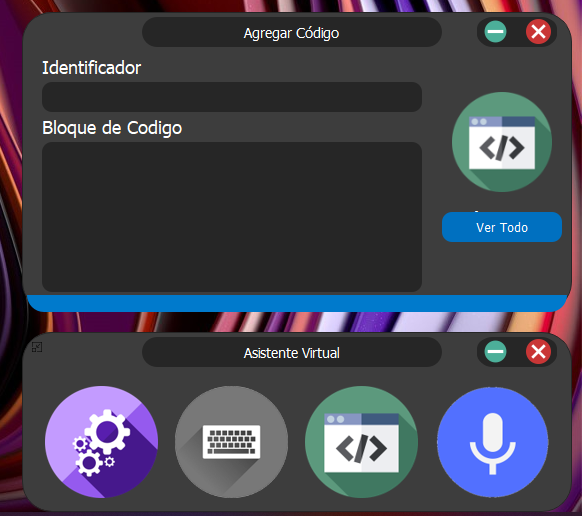
****

Ilustración Modulo: Agregar Código Local

1. Entrada de texto**:** La búsqueda por texto abre un menú parecido al de búsqueda por voz, con la diferencia de que nosotros debemos de ingresar las peticiones tal como se muestra en la ilustración 3, una vez que se haga la búsqueda obtendremos una ventana más que nos da diferentes opciones:

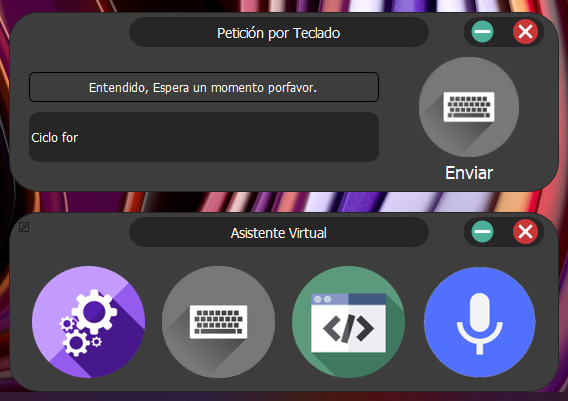
****

Ilustración Modulo: Búsqueda por teclado.

* 1. Resultados de Búsqueda**:** La ilustración 4 muestra como es que se presentan los resultados de las peticiones tanto de las peticiones por voz como por teclado, encontraremos funciones como siguiente resultado de la página (flechas azules, esquina superior derecha), fuente, siguiente página y guardar el resultado.

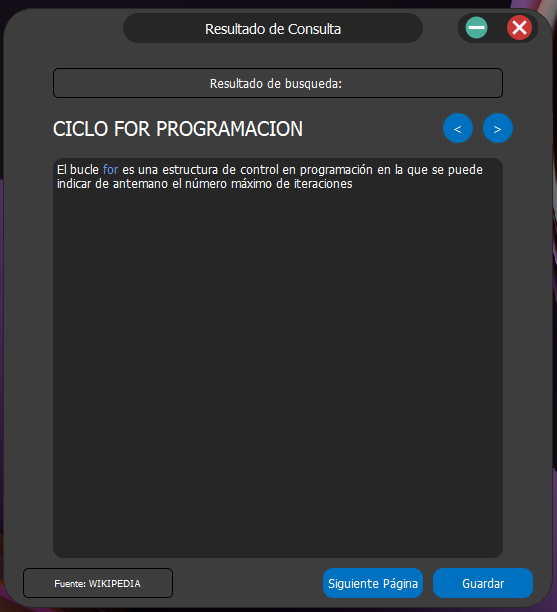
****

Ilustración Resultados de la consulta.

1. Ajustes**:** En el apartado de ajustes tenemos diferentes opciones como: velocidad del habla, que regula la velocidad con la que el asistente interactúa con nosotros; La respuesta de asistente por voz, que es la opción que nos permite apagar la interacción por voz del asistente hacia nosotros; Tema de la interfaz, que nos permite seleccionar entre claro y obscuro; Asistente de aplicación, que nos permite seleccionar la entonación, acento y género del asistente, de acuerdo a los paquetes de recursos que **ya tengamos instalados previamente**, finalmente, Dispositivo de entrada de audio, que nos permite seleccionar entre los diferentes dispositivos que tengamos conectados al ordenador. Todas esas opciones podemos observarlas en la ilustración 5.

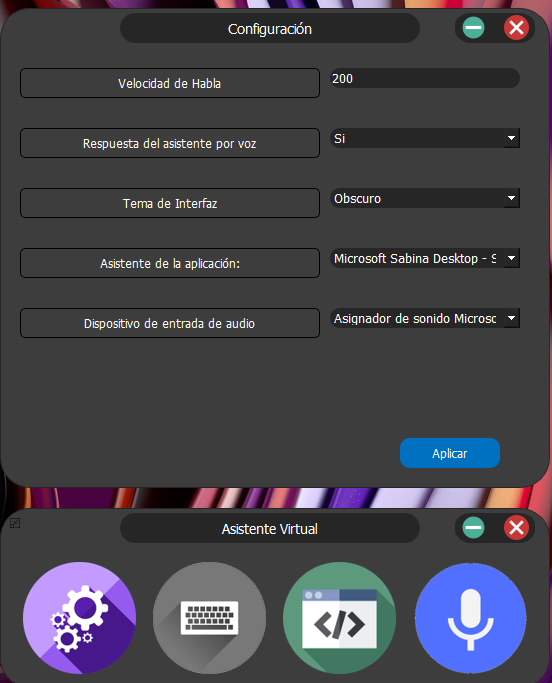
****

Ilustración Modulo de configuración